

**PEDECIBA Informática**  
**Instituto de Computación – Facultad de Ingeniería**  
**Universidad de la República**  
**Montevideo, Uruguay**

---

# **Tesis de Doctorado**

## **en Informática**

---

**MINERVA: Model driven and Service  
oriented framework for the continuous  
improvement of business process & related  
tools**

**Andrea Delgado**

**Supervisor:** Dr. Alberto Pardo

**Orientador:** Dr. Francisco Ruiz

**Co-orientador:** Dr. Ignacio García-Rodríguez de Guzmán

**Revisores**

Dr. Manfred Reichert, University of Ulm, Ulm, Germany

Dr. Pericles Loucopoulos, Loughborough University, Leicestershire, United Kingdom

**Tribunal**

Dr. Mario Piattini, Universidad de Castilla - La Mancha, Ciudad Real, España

Dra. Angeles Saavedra Places, Universidad de La Coruña, La Coruña, España

Dra. Barbara Weber, University of Innsbruck, Innsbruck, Austria

Dra. Valeria de Castro, Universidad Rey Juan Carlos, Madrid, España

Dr. Antonio Vallecillo, Universidad de Málaga, Málaga, España

**Marzo 2012**

Minerva : model driven and service oriented framework for the continuous improvement of business process & related tools

Delgado, Andrea

ISSN 0797-6410

Tesis de Doctorado en Informática

Reporte Técnico RT 12-03

PEDECIBA

Instituto de Computación – Facultad de Ingeniería

Universidad de la República.

Montevideo, Uruguay, marzo de 2012

To my family,  
who have always been there for me.



# Acknowledgments

In the first place, I want to express deep thanks to my supervisors Dr. Francisco Ruiz and Dr. Ignacio García-Rodríguez de Guzmán, for giving me of their knowledge, which is extensive, as well as providing me with the always wise and accurate directions for the work, along with the support needed to carry out this thesis. It has been a real pleasure to work with you over these last years. In particular, I want to thank Paco in whom I have also found a very good friend.

I want to extend my thanks to all the Alarcos Research Group too, for having me here these four years; specially to Dr. Mario Piattini for this great opportunity and to Dr. Félix García for the joint work on business process topics. I can't name everyone here, and I don't want to forget anyone, so my thanks goes to all the Alarcos professors, lecturers and technicians I have met, with whom I have had the chance to share this time.

I specially want to thank my PhD companions with whom I worked, first in Indra and afterwards in the ITSI: César P., Laura, Ana, Ricardo, Pedro, Tomás, Carlos, and my compatriots Bea and Fede with whom I shared many "mates", and to my former PhD colleagues: César G., Francisco, Bea M., Gaby, Andrés and Elvira, whom have finished their PhD in the years I have been here; you have been a great group to share time with.

And thanks to all of you for the "cañas" and typical meals shared at social gatherings and in leisure time. Sharing these years with all of you have been a great experience in my life, in more than one sense, the academic, the human, and the cultural ones.

I also want to thank the Quality Engineering Group of the University of Innsbruck, Austria, for giving me the opportunity to carry out my research stay with them, to the head of the group Dr. Ruth Breu, to Dr. Michael Felderer and to Dr. Barbara Weber for giving me such good advice.

I am also grateful to the Computer Science Institute (InCo) of the University of the Republic for giving me the permission to come to the University of Castilla - La Mancha for my PhD studies, as well as to the ALFA LERnet program from the European Union for giving me the grant to attend my first year of PhD courses, and the Agencia Nacional de Investigación e Innovación (ANII) from the Uruguayan government for the grant allowing me to finish my PhD studies in Spain.

Special thanks goes to Daniel Calejari the coordinator of the COAL group at the InCo, for believing in me and for giving me so many opportunities to contribute to the great work he is doing. The same applies to Dr. Alberto Pardo my supervisor at the InCo, my thanks to him for giving me the opportunity to apply to the ALFA LERnet program he supervised, and for the support in the many activities of the thesis. I also want to express my gratitude to the student groups that carried out the evaluation of tools for my thesis, as well as the development of the new tools that I have defined.

To my Uruguayan friends, who have been a great support in the distance and every time I went back to "el paisito", to all of them my warmest thank you, but especially to Pablin, who is also my brother.

To the many friends I have made while living in Ciudad Real, for making me feel at home and for opening their homes to me, I hope I can return the hospitality some day.

Last but not least -on the contrary, the most important thanks of all- goes to my family, who have always been there for me: my parents Alicia y Luis, who have by example made me who I am, my sisters Marcela, Ana and Patricia, my brother Rodrigo, with whom sharing life has always been the best thing in the world, to Leo, Cano and Fede, who are also my brothers; they have made my family even better, if that is possible. To the unforgettable Chola, Malena and Teresa for being part of our family. And finally, to the sunshine of my life, my pride and joy, my nieces: Lucía, Chiara and Violeta and my nephews León, Luciano and Manuel. I love you all very much!



*The important thing is not to stop questioning.*

Albert Einstein





# Contents



# Table of Contents

<b>Acknowledgments</b>	<b>i</b>
<b>Contents</b>	<b>v</b>
<b>Table of Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xxv</b>
<b>Abstract</b>	<b>xxvii</b>
<b>Resumen</b>	<b>xxix</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Hypothesis and objectives . . . . .	3
1.3. Context . . . . .	4
1.3.1. Research Groups . . . . .	4
1.3.2. R&D projects . . . . .	4
1.3.2.1. INGENIO . . . . .	5
1.3.2.2. INGENIOSO . . . . .	6
1.3.2.3. COMPETISOFT . . . . .	6
1.3.2.4. ALTAMIRA . . . . .	7
1.3.2.5. PEGASO/MAGO . . . . .	7
1.4. Document structure . . . . .	8
<b>2. Research methods</b>	<b>11</b>
2.1. Research methods in Software Engineering . . . . .	11
2.1.1. Action-Research . . . . .	12
2.1.2. Empirical Software Engineering . . . . .	14
2.1.2.1. Experiments . . . . .	14
2.1.2.2. Case study . . . . .	17
2.1.3. Systematic review . . . . .	20
2.2. Use of research methods in this thesis . . . . .	23
2.2.1. Action-Research . . . . .	23

2.2.2.	Experiments . . . . .	24
2.2.3.	Case study . . . . .	24
2.2.4.	Systematic review . . . . .	25
2.3.	Conclusions . . . . .	25
<b>3.</b>	<b>State of the art</b>	<b>27</b>
3.1.	Business Process Management (BPM) . . . . .	27
3.1.1.	BPM and BPMS . . . . .	27
3.1.2.	BP Concepts and lifecycle . . . . .	30
3.1.2.1.	BP definitions . . . . .	30
3.1.2.2.	BP patterns . . . . .	31
3.1.2.3.	BP lifecycle . . . . .	32
3.1.3.	Main standards for BPs . . . . .	34
3.1.3.1.	Business Process Model and Notation (BPMN2) . . . . .	34
3.1.3.2.	XML Process Definition Language (XPDL) . . . . .	40
3.1.3.3.	Web Services BP Execution Language (WS-BPEL) . . . . .	42
3.2.	Service Oriented Computing (SOC) . . . . .	45
3.2.1.	SOC, SOE, SOD and SOA . . . . .	45
3.2.2.	Service Concepts . . . . .	47
3.2.2.1.	Service definitions . . . . .	48
3.2.2.2.	Service design principles . . . . .	50
3.2.3.	Main standards for service orientation . . . . .	51
3.2.3.1.	Service Oriented Architecture Modeling (SoaML) . . . . .	52
3.3.	Model Driven Development (MDD) . . . . .	55
3.3.1.	MDE, MDD and MDA . . . . .	55
3.3.2.	Models, Metamodels and Transformations . . . . .	57
3.3.3.	UML Profiles vs. DSL . . . . .	59
3.3.4.	Main standards for model-driven . . . . .	61
3.3.4.1.	Model Driven Architecture (MDA) . . . . .	61
3.3.4.2.	Meta-Object Facility (MOF) . . . . .	62
3.3.4.3.	Query/Views/Transformations Language (QVT) . . . . .	63
3.4.	Continuous Process Improvement (CPI) . . . . .	65
3.4.1.	BP improvement concepts and types . . . . .	65
3.4.2.	BP execution measurement . . . . .	67
3.4.2.1.	Activity instances lifecycle and execution times . . . . .	67
3.4.2.2.	Control flow view on execution times . . . . .	68
3.4.2.3.	Time and Cost execution measures . . . . .	69
3.4.2.4.	BP execution event logs . . . . .	69
3.4.3.	Main standards for CPI . . . . .	71

3.4.3.1.	Business Process Maturity Model (BPMM)	71
3.5.	Systematic literature review	73
3.5.1.	Research Question, Search String and Sources	74
3.5.2.	Study Selection and Information Extraction	74
3.5.3.	Analysis of the Results	75
3.5.3.1.	Summary of studies	76
3.5.3.2.	Main Principles in Paradigm Integration	77
3.5.3.3.	Summary of Main Principles and Selected Studies	84
3.6.	Conclusions	86
<b>4.</b>	<b>MINERVA framework</b>	<b>87</b>
4.1.	Motivation	87
4.2.	MINERVA definition	88
4.3.	Dimensions view	90
4.3.1.	Conceptual dimension	90
4.3.1.1.	Conceptual Architecture	90
4.3.1.2.	Metamodels	91
4.3.1.3.	Ontologies	92
4.3.2.	Methodological dimension	106
4.3.3.	Tools support dimension	107
4.4.	Process view	108
4.5.	Conclusions	111
<b>5.</b>	<b>Business Process Continuous Improvement Process (BPCIP)</b>	<b>113</b>
5.1.	Introduction	113
5.2.	BPCIP Disciplines	114
5.2.1.	Business Modeling Discipline	115
5.2.1.1.	BM1 - Assess the Organization	115
5.2.1.2.	BM2 - Identify Business Processes	116
5.2.1.3.	BM3 - Redesign Business Processes	116
5.2.2.	Business Process Validation & Verification Discipline	117
5.2.2.1.	VV1 - Validate Business Processes	117
5.2.2.2.	VV2 - Verify Business Processes	118
5.2.3.	Business Process Implementation Discipline	118
5.2.3.1.	I1 - Implement BPs with services	119
5.2.3.2.	I2 - Reimplement services	120
5.2.4.	Business Process Analysis Discipline	120
5.2.4.1.	A1 - Monitor BPs execution	121
5.2.4.2.	A2 - Analyze BPs execution	121

5.2.5.	Business Process Execution Measurement Discipline . . . . .	122
5.2.5.1.	EM1 - Select execution measures . . . . .	122
5.2.5.2.	EM2 - Implement execution measures collection . . . . .	123
5.2.5.3.	EM3 - Collect execution measures . . . . .	123
5.2.5.4.	EM4 - Analyze execution measurement results . . . . .	123
5.2.6.	Business Process Improvement Discipline . . . . .	124
5.2.6.1.	IM1 - Define improvements . . . . .	124
5.2.6.2.	IM2 - Diagnose Processes . . . . .	125
5.2.6.3.	IM3 - Formulate improvements . . . . .	125
5.2.6.4.	IM4 - Assess improvement effort . . . . .	126
5.3.	BPCIP Phases . . . . .	126
5.3.1.	Design&Analysis phase . . . . .	127
5.3.2.	Configuration phase . . . . .	128
5.3.3.	Enactment phase . . . . .	128
5.3.4.	Evaluation phase . . . . .	129
5.4.	EPF implementation . . . . .	129
5.4.1.	BPCIP method plug-in . . . . .	130
5.4.2.	BPCIP web site . . . . .	130
5.5.	Conclusions . . . . .	131
<b>6.</b>	<b>Business Process Execution Measurement Model (BPEMM)</b>	<b>133</b>
6.1.	Introduction . . . . .	133
6.2.	BPEMM definition . . . . .	135
6.2.1.	Execution measures specification . . . . .	135
6.2.2.	Views, dimensions and hierarchy . . . . .	137
6.2.3.	Assumptions for calculations . . . . .	139
6.3.	Execution measures . . . . .	140
6.3.1.	Generic BP execution view . . . . .	140
6.3.1.1.	Time dimension . . . . .	140
6.3.1.2.	Cost dimension . . . . .	142
6.3.1.3.	Quality dimension . . . . .	143
6.3.1.4.	Flexibility dimension . . . . .	144
6.3.2.	Lean execution view . . . . .	145
6.3.2.1.	Time dimension . . . . .	145
6.3.2.2.	Quality dimension . . . . .	146
6.3.3.	Services execution view . . . . .	146
6.3.3.1.	Time dimension . . . . .	146
6.3.3.2.	Quality dimension . . . . .	149
6.4.	Example of application . . . . .	150
6.4.1.	Calculations for the BP generic view . . . . .	150
6.5.	Conclusions . . . . .	157

<b>7. Business Process Service Oriented Methodology (BPSOM)</b>	<b>159</b>
7.1. Introduction . . . . .	159
7.2. BPSOM Disciplines . . . . .	162
7.2.1. Business Modeling Discipline . . . . .	162
7.2.1.1. BM1 - Assess the target Organization . . . . .	163
7.2.1.2. BM2 - Identify Business Processes . . . . .	163
7.2.2. Design Discipline . . . . .	165
7.2.2.1. D1 - Identify and categorize services . . . . .	166
7.2.2.2. D2 - Specify services . . . . .	169
7.2.2.3. D3 - Investigate existing services . . . . .	172
7.2.2.4. D4 - Assign components to services . . . . .	173
7.2.2.5. D5 - Define services interaction . . . . .	174
7.2.3. Implementation Discipline . . . . .	175
7.2.3.1. I1 - Implement services . . . . .	175
7.3. BPSOM phases . . . . .	176
7.3.1. Inception phase . . . . .	177
7.3.2. Elaboration phase . . . . .	178
7.3.3. Construction phase . . . . .	179
7.3.4. Transition phase . . . . .	179
7.4. EPF implementation . . . . .	179
7.4.1. BPSOM method plug-in . . . . .	179
7.4.2. BPSOM web site . . . . .	180
7.5. Conclusions . . . . .	181
<b>8. Generation of SoaML models from BPMN2 models</b>	<b>183</b>
8.1. Introduction . . . . .	183
8.2. Integration in BPSOM . . . . .	185
8.3. BPMN2 vs SoaML correspondences . . . . .	188
8.3.1. BPMN2 key elements in transformations . . . . .	188
8.3.2. SoaML key elements in transformations . . . . .	189
8.3.3. BPMN2 vs SoaML correspondences definition . . . . .	189
8.4. QVT transformations . . . . .	191
8.4.1. General definitions . . . . .	191
8.4.2. QVT relations defined . . . . .	193
8.4.2.1. Model, Participants and Messages rules . . . . .	194
8.4.2.2. Services rules . . . . .	195
8.4.2.3. Ports and MessageType update rules . . . . .	197
8.4.2.4. ServicesArchitecture rules . . . . .	198
8.4.2.5. QVT transformations code . . . . .	200

8.4.3.	Empirical evaluation . . . . .	200
8.5.	Example of application . . . . .	202
8.5.1.	Application of the generation procedure . . . . .	204
8.5.2.	Generated SoaML diagrams . . . . .	205
8.5.2.1.	ServiceArchitecture diagram . . . . .	205
8.5.2.2.	Services Interfaces and ServiceContracts . . . . .	206
8.5.2.3.	Participants . . . . .	207
8.5.2.4.	MessageTypes . . . . .	208
8.6.	Conclusions . . . . .	209
<b>9.</b>	<b>Tools support</b>	<b>211</b>
9.1.	Introduction . . . . .	211
9.2.	BPCIP phases tools support . . . . .	212
9.2.1.	Tools decision criteria . . . . .	212
9.2.2.	Design&Analysis phase . . . . .	213
9.2.2.1.	BP modeling . . . . .	213
9.2.2.2.	BPEMM measures specification . . . . .	215
9.2.3.	Configuration phase . . . . .	216
9.2.3.1.	Eclipse SoaML plug-in . . . . .	219
9.2.3.2.	Eclipse iS4BPe plug-in . . . . .	223
9.2.4.	Enactment phase . . . . .	225
9.2.5.	Evaluation phase . . . . .	228
9.2.5.1.	ProM BPEMM plug-in . . . . .	228
9.2.6.	BPCIP lifecycle guidance tool support . . . . .	231
9.3.	Conclusions . . . . .	232
<b>10.</b>	<b>MINERVA framework validation</b>	<b>233</b>
10.1.	Introduction . . . . .	233
10.2.	Empirical validation of QVT transformations . . . . .	236
10.2.1.	Problem definition . . . . .	236
10.2.2.	Planning of the experiment . . . . .	237
10.2.2.1.	Context selection . . . . .	237
10.2.2.2.	Subjects selection . . . . .	238
10.2.2.3.	Variables selection . . . . .	238
10.2.2.4.	Formulation of hypothesis . . . . .	238
10.2.2.5.	Design of the experiment . . . . .	240
10.2.2.6.	Experimental materials . . . . .	241
10.2.2.7.	Assessment of validity . . . . .	244
10.2.3.	Operation of the experiment . . . . .	245



10.2.3.1. Preparation . . . . .	245
10.2.3.2. Execution . . . . .	245
10.2.3.3. Data validation . . . . .	246
10.2.4. Data analysis and interpretation of results . . . . .	246
10.2.4.1. Descriptive statistics . . . . .	246
10.2.4.2. Hypothesis testing . . . . .	249
10.2.5. Presentation and dissemination . . . . .	251
10.3. Empirical validation of BPSOM . . . . .	251
10.3.1. Background . . . . .	252
10.3.2. Design . . . . .	252
10.3.3. Case selection . . . . .	253
10.3.4. Procedures and roles . . . . .	253
10.3.5. Data collection . . . . .	253
10.3.6. Analysis . . . . .	254
10.3.7. Validity . . . . .	256
10.3.8. BPSOM use in the case study . . . . .	256
10.3.8.1. Selling products on a mobile commercial platform . . . . .	256
10.3.8.2. BPSOM application and tool support . . . . .	257
10.3.9. Conclusions and lessons learned . . . . .	259
10.4. Empirical validation of BPCIP . . . . .	259
10.4.1. Background . . . . .	260
10.4.2. Design . . . . .	260
10.4.3. Case selection . . . . .	261
10.4.4. Procedures and roles . . . . .	261
10.4.5. Data collection . . . . .	262
10.4.6. Analysis . . . . .	262
10.4.7. Validity . . . . .	263
10.4.8. BPCIP use in the case study . . . . .	264
10.4.8.1. Patient MAS in HGCR . . . . .	264
10.4.8.2. BPCIP application and tool support . . . . .	264
10.4.8.3. Analysis with ProM . . . . .	273
10.4.9. Conclusions and lessons learned . . . . .	277
10.5. Conclusions . . . . .	277

<b>11. Conclusions and future work</b>	<b>279</b>
11.1. Attainment of objectives	279
11.2. Results that support this thesis	281
11.2.1. Journal articles (JCR)	283
11.2.2. Journals articles (other)	283
11.2.3. Book Chapters	283
11.2.4. Conferences Level A	283
11.2.5. Conferences Level B	283
11.2.6. Other Conferences	284
11.2.7. Workshops	284
11.3. Main research contributions	285
11.4. Open research lines and future work	286
<b>Appendices</b>	<b>289</b>
<b>A. Data extraction from primary studies of the systematic review</b>	<b>291</b>
A.1. Overview	291
A.2. Data extraction tables	291
<b>B. BPCIP and BPSOM Web Sites (implemented with EPF Composer)</b>	<b>303</b>
B.1. Overview	303
B.2. BPCIP Web Site	303
B.2.1. Disciplines	304
B.2.1.1. Business Modeling Discipline	304
B.2.1.2. BP Validation and Verification Discipline	305
B.2.1.3. BP Implementation Discipline	305
B.2.1.4. BP Analysis Discipline	306
B.2.1.5. BP Execution Measurement Discipline	306
B.2.1.6. BP Improvement Discipline	307
B.2.2. Work products	307
B.2.3. Roles	308
B.2.4. Lifecycle	308
B.3. BPSOM Web Site	309
B.3.1. Disciplines	309
B.3.1.1. Business Modeling Discipline	310
B.3.1.2. Design Discipline	311
B.3.1.3. Implementation Discipline	312
B.3.2. Work products	312
B.3.3. Roles	313
B.3.4. Lifecycle	313

<b>C. QVT transformations code</b>	<b>315</b>
C.1. Overview . . . . .	315
C.2. Generation from ServiceTask . . . . .	315
C.2.1. Bidirectional option with simple UML Interfaces . . . . .	315
C.2.2. Bidirectional option with ServiceInterfaces . . . . .	321
C.2.3. Unidirectional option with simple UML Interfaces . . . . .	327
C.3. Generation from ServiceTask and other elements . . . . .	333
C.3.1. Bidirectional option with simple UML Interfaces . . . . .	333
C.3.2. Bidirectional option with ServiceInterfaces . . . . .	340
C.3.3. Unidirectional option with simple UML Interfaces . . . . .	346
<b>D. Experimental material of the QVT transformations validation experiment</b>	<b>353</b>
D.1. Overview . . . . .	353
D.2. Experimental materials . . . . .	353
D.2.1. Tutorial . . . . .	353
D.2.2. Part 1 . . . . .	359
D.2.3. Part 2 . . . . .	377
<b>E. HGCR case study implementation in XPDL and WS-BPEL and simulation of resources</b>	<b>389</b>
E.1. Overview . . . . .	389
E.2. Implementation and execution in Bonita . . . . .	389
E.3. Implementation and execution in Intalio . . . . .	393
E.4. Alternative modeling of resources in CPN Tools . . . . .	397
<b>Bibliography</b>	<b>399</b>
<b>Acronyms</b>	<b>409</b>



# List of Figures

1.1. Vertical and horizontal visions for business processes realization based on [Erl, 2005]	2
1.2. R&D projects context of this thesis work	5
2.1. Action-research cyclic nature	13
2.2. Stages in carrying out experiments from [Wohlin et al., 2000]	15
2.3. Decision tree for analysis techniques from (Pfleeger, 1994)	16
2.4. Stages in case study development from [Yin, 2002]	18
2.5. Three stages for systematic reviews defined by [Kitchenham and Charters, 2007]	21
2.6. Application of Action-Research in this thesis work	23
3.1. Evolution of BPM technology from [ter Hofstede et al., 2009]	28
3.2. (a) Value chain and (b) value system for organization (E) from [Weske, 2007]	29
3.3. Organizational-level BPM from [Weske, 2007] based on Schmelzer and Seselmann (2006)	29
3.4. BP model and BP instances based on [Weske, 2007]	31
3.5. (a) orchestration and (b) collaborative BP from [Weske, 2007]	31
3.6. Multiple choice control flow pattern (a) Petri Nets (b) BPMN and (c) UML AD	32
3.7. BP lifecycle from [Weske, 2007]	33
3.8. BPMN standard evolution from [Rademakers and van Liempd, 2011-2012]	34
3.9. BPMN2 collaborative BP example from the BPMN2 poster of BPMBerlin	35
3.10. BPMN2 elements (some) from the BPMN2 poster of BPMBerlin	36
3.11. BPMN2 layers structure and BPMN Core elements from [OMG, 2011a]	37
3.12. BPMN2 XML file example	37
3.13. BPMN2 metamodel for Collaborations from [OMG, 2011a]	38
3.14. BPMN2 XML file for WS invocation from ServiceTaks	39
3.15. BPMN2 XML file with diagram information	39
3.16. XPDL standard evolution with BPMN adapted from [Shapiro and Gagne, 2010]	40
3.17. WfMC reference architecture from [Havey, 2005]	40
3.18. XPDL process definition metamodel from [Shapiro and Gagne, 2010]	41
3.19. XPDL file example from [WfMC, 2008]	42
3.20. WS-BPEL standard evolution from [Rademakers and van Liempd, 2011-2012]	42
3.21. WS-BPEL process execution from [Havey, 2005]	43
3.22. WS-BPEL metamodel from [Hornung et al., 2006]	44

3.23. WS-BPEL example file from [Havey, 2005]	44
3.24. Extended SOA layers from [Papazoglou et al., 2007]	46
3.25. Find-bind-invoke paradigm in SOA from [Endrei et al., 2004]	47
3.26. SOC, SOE, SOD and SOA relationships	47
3.27. Services logic encapsulation from [Erl, 2005]	48
3.28. (a) unidirectional and (b) bidirectional service definition based on interfaces	49
3.29. Services classification according to [Erl, 2005]	50
3.30. SoaML ServicesArchitecture and Participants specification from [OMG, 2009b]	53
3.31. SoaML ServiceContract, Interfaces and ServiceInterfaces specification from [OMG, 2009b]	53
3.32. Some stereotypes defined in the SoaML profile from [OMG, 2009b]	54
3.33. Some elements defined in the SoaML metamodel from [OMG, 2009b]	54
3.34. MDE scope, standards and tools from [Bézivin, 2004]	56
3.35. MDE, MDD and MDA relationships	57
3.36. Relationships between system-model-metamodel from [Bézivin, 2004]	58
3.37. Model transformation adapted from [Bézivin, 2005]	58
3.38. MOF and UML use for defining new languages from [Vallecillo, 2000-2011]	59
3.39. UML profile definition (a) (b) and use (c) example adapted from [Vallecillo, 2000-2011]	60
3.40. Four modeling layers of MOF from [Kleppe et al., 2003]	63
3.41. Relationships between QVT metamodels from [OMG, 2008c]	63
3.42. QVTBase package - transformations and rules from [OMG, 2008c]	64
3.43. QVT Relations package from [OMG, 2008c]	65
3.44. CPI and BPR differences (a) impact of changes, (b) changes over time from [van der Aalst, 2002]	66
3.45. (a) Activity lifecycle and (b) execution times from [zur Muehlen, 2004]	67
3.46. Use of Gateways and formulas to calculate times [Laguna and Marklund, 2005]	68
3.47. Process log XML format (a) and transactional model (b) for event types from [van Dongen et al., 2005]	70
3.48. An example of an MXML event log from examples of [van der Aalst, 2011]	71
3.49. BPMM maturity levels from [OMG, 2008b]	72
3.50. Definition of Process Areas in BPMM from [OMG, 2008b]	73
3.51. Publications by Paradigm integration (a) and Type of case study (b)	76
3.52. Business Process (a) and Service/Software (b) notations used	76
3.53. Main principles in paradigms integration	77
3.54. Main principles in selected studies by paradigms integration	84
4.1. MINERVA framework views	89
4.2. Conceptual Dimension elements	90
4.3. Ontology and sub-ontologies for BP lifecycle realized by services	93

4.4. BP Modeling sub-Ontology (BPMsO) diagram . . . . .	96
4.5. Service Oriented Modeling sub-Ontology (SOMsO) diagram . . . . .	99
4.6. BP Simulation sub-Ontology (BPSsO) diagram . . . . .	100
4.7. BP Execution sub-Ontology (BPEsO) diagram . . . . .	102
4.8. Service Oriented Execution sub-Ontology (SOEsO) diagram . . . . .	102
4.9. BP Evaluation sub-Ontology (BPEVsO) . . . . .	103
4.10. UML class diagram for BPMsO and SOMsO integration . . . . .	104
4.11. Methodological dimension elements . . . . .	106
4.12. Tool support dimension elements definition . . . . .	107
4.13. MINERVA framework lifecycle . . . . .	108
4.14. MINERVA method of work through BPCIP . . . . .	109
5.1. MINERVA method of work through BPCIP . . . . .	114
5.2. Business Modeling activity detail diagram . . . . .	115
5.3. BP validation and verification activity detail diagram . . . . .	117
5.4. BP implementation activity detail diagram . . . . .	119
5.5. BP Analysis activity detail diagram . . . . .	121
5.6. BP Execution Measurement activity detail diagram . . . . .	122
5.7. BP Improvement Discipline activity detail diagram . . . . .	124
5.8. BPCIP Design&Analysis phase activity diagram . . . . .	127
5.9. BPCIP Configuration phase activity diagram . . . . .	128
5.10. BPCIP Enactment phase activity diagram . . . . .	128
5.11. BPCIP Evaluation phase activity diagram . . . . .	129
5.12. Example of BPCIP method plug-in definition in EPF Composer . . . . .	130
5.13. Global view of BPSOM web site created using EPF composer . . . . .	131
6.1. BPEMM use in BPCIP . . . . .	133
6.2. MINERVA framework method of work . . . . .	134
6.3. Dimensions of the devil's quadrangle [Brand and van der Kolk, 1995, Reijers, 2003]	137
6.4. Hierarchy of execution measures defined in BPEMM . . . . .	138
6.5. Cube view of the execution measures defined in BPEMM . . . . .	138
6.6. Defined times for activities and BP instances execution . . . . .	140
6.7. Defined times for services execution and BP activities . . . . .	147
6.8. "Patient Admission and Registration for Major Ambulatory Surgery (MAS)" in BPMN2 . . . . .	151
6.9. Example BP case for the Patient MAS BP with execution times . . . . .	153
6.10. Example BP case for the Patient MAS BP with BP cases for branch . . . . .	155
7.1. How BPSOM is added to the existing software development process . . . . .	160
7.2. BPSOM activities flow as a BPMN2 model . . . . .	161

7.3. MINERVA method of work through BPCIP . . . . .	161
7.4. Business Modeling activity detailed diagram . . . . .	163
7.5. Patient MAS business process in BPMN2 . . . . .	165
7.6. Design Discipline activity detailed diagram . . . . .	166
7.7. SoaML ServicesArchitecture diagram for the PatientMAS BP . . . . .	168
7.8. SoaML Participants and Services diagram with Ports . . . . .	168
7.9. SoaML Interfaces diagram for the defined services . . . . .	170
7.10. SoaML MessageTypes diagram for the defined services . . . . .	170
7.11. SoaML ServiceContracts diagram for the defined services . . . . .	171
7.12. SoaML choreography diagram associated with the ServiceContract . . . . .	171
7.13. SoaML component reuse for the “ReceiveRequestForAppointment” service . . . . .	173
7.14. SoaML component definition for the “ReceivePatientMedicalrecord” service . . . . .	174
7.15. UML sequence diagram showing defined services interaction . . . . .	175
7.16. Implementation Discipline activity detailed diagram . . . . .	176
7.17. Relationship between BPCIP and BPSOM phases . . . . .	177
7.18. BPSOM Inception phase activity diagram . . . . .	178
7.19. BPSOM Elaboration phase activity diagram . . . . .	178
7.20. BPSOM Construction phase activity diagram . . . . .	179
7.21. Example of BPSOM method plug-in definition in EPF Composer . . . . .	180
7.22. Global view of BPSOM web site created using EPF composer . . . . .	181
8.1. MDA vision in MINERVA framework . . . . .	183
8.2. Business processes and services transformations vision of MINERVA . . . . .	184
8.3. MINERVA method of work through BPCIP . . . . .	185
8.4. Model-driven approach integrated in the BPSOM methodology . . . . .	186
8.5. Step by step generation procedure and tool support . . . . .	187
8.6. Service design generation options provided in BPSOM . . . . .	187
8.7. BPMN2 [OMG, 2011a] metamodel elements used in the correspondences to SoaML	188
8.8. SoaML [OMG, 2009b] metamodel elements used in the correspondences to BPMN2	189
8.9. Key correspondences between BPMN2 and SoaML metamodels for ServiceTask . .	190
8.10. Key correspondences between BPMN2 and SoaML metamodels for ServiceTask and other elements . . . . .	191
8.11. General Algorithm for services generation pseudo code . . . . .	192
8.12. Hierarchy and dependencies between the QVT relations defined . . . . .	193
8.13. QVT relations for generating: (a) Model and (b) Participants in QVT graphical form	194
8.14. QVT relations for generating MessageTypes in QVT graphical form . . . . .	195
8.15. QVT top relations for services generation (a) from ServiceTask and (b) with In- terface, Operation and Message for the service provider, in QVT graphical form . . . . .	196
8.16. QVT relation for services generation from ServiceTask in QVT graphical form . .	197



8.17. QVT relations for Participants Ports update in QVT graphical form . . . . .	198
8.18. QVT relations for the ServicesArchitecture generation in QVT graphical form . .	199
8.19. Elements in BPMN2 and SoaML models . . . . .	201
8.20. Generation times for BPMN2 model and generation option . . . . .	202
8.21. Patient MAS from the HGCR in BPMN2 . . . . .	203
8.22. Generated SoaML XMI file in Eclipse core editor showing the UML base model: (a) SoaML model general structure, (b) Service specification . . . . .	204
8.23. Generated SoaML XMI file stereotypes application (a) Eclipse ecore editor (b) XML view . . . . .	205
8.24. Generated ServicesArchitecture diagram . . . . .	206
8.25. Generated services specification bidirectional option . . . . .	206
8.26. Generated services specification unidirectional option . . . . .	207
8.27. Generated services specification ServiceInterface bidirectional option . . . . .	207
8.28. Generated Participants and Ports bidirectional option . . . . .	208
8.29. Generated MessageTypes bidirectional option . . . . .	208
9.1. MINERVA method of work through BPCIP . . . . .	212
9.2. SMTTool example of BPEMM execution measures specification in graphical form .	216
9.3. Tools support for BP and services modeling and implementation . . . . .	217
9.4. Eclipse MINERVA design distribution screenshot . . . . .	219
9.5. Eclipse general Architecture with Papyrus and SoaML plug-ins . . . . .	220
9.6. Subsystems of the SoaML Extended Papyrus Architecture . . . . .	220
9.7. Components view of the SoaML solution . . . . .	221
9.8. Eclipse SoaML plug-in Import/Export and visualization with EMF . . . . .	222
9.9. Eclipse SoaML plug-in new ServicesArchitecture diagram layout . . . . .	222
9.10. Eclipse SoaML plug-in population of the ServicesArchitecture diagram . . . . .	223
9.11. Web Services Invocation generated for the Activiti engine . . . . .	224
9.12. ServiceTaks inserted information for the Activiti engine . . . . .	224
9.13. ProM BPEMM plug-in definitions . . . . .	229
9.14. XMLSchema (1) and example (2) of BP Configuration file ProM BPEMM plug-in	229
9.15. ProM BPEMM plug-in example for BP cases option . . . . .	230
9.16. ProM BPEMM plug-in example for each BP case option . . . . .	230
9.17. ProM BPEMM plug-in example for an activity through all BP cases . . . . .	231
10.1. MINERVA method of work coverage by the case study defined initially . . . . .	234
10.2. MINERVA method of work coverage by the two case studies finally defined . . . .	235
10.3. Summary of the experimental plan . . . . .	237
10.4. Part 1 example of diagrams exercise for Suitability . . . . .	242
10.5. Part 1 example of textual correspondence rules exercise for Suitability . . . . .	243
10.6. Part 2 example of exercises for Understandability . . . . .	244

10.7. Average times for Suitability and Understandability . . . . .	247
10.8. Agreements per presentation type and model for Suitability . . . . .	248
10.9. Correct answers and evaluation per model for Understandability . . . . .	248
10.10Agreements and Correct Answers per SoaML diagram . . . . .	249
10.11Maximums and minimums of answers given in the questionnaire . . . . .	255
10.12General view of the case study BP based on participants interactions . . . . .	257
10.13Generated SoaML Service Architecture diagram . . . . .	258
10.14Generated (some) Service Contracts and Interfaces . . . . .	258
10.15Generated (some) Service Interfaces diagrams . . . . .	258
10.16Complete Patient MAS BP from HGCR . . . . .	264
10.17Example of User forms defined in Activiti . . . . .	265
10.18Patient MAS sub-processes modeled in Activiti . . . . .	266
10.19Example of task lists assigned to roles in Activiti . . . . .	267
10.20Example of ServiceTask implementation as sending mail in Activiti . . . . .	267
10.21Example queries to obtain the execution data from Activiti . . . . .	268
10.22Fluxicon transformation of Activiti .csv log into MXML . . . . .	268
10.23Activiti MXML log loaded into ProM . . . . .	268
10.24Global view of the Patient MAS Petri Net defined in CPNTools . . . . .	269
10.25Adapted Patient MAS in BPMN2 modeled in Oryx . . . . .	270
10.26Hospital sub-page showing the transitions substitution modeling . . . . .	271
10.27Top page environment for resources modeling with central fusion place . . . . .	271
10.28Resources modeling example for activity “Check preconditions for MAS” . . . . .	272
10.29Service sub-page with a queuing approach . . . . .	272
10.30Summary of the event log from CPNTools loaded into ProM . . . . .	273
10.31Inspector option of ProM showing the execution of BP instances . . . . .	273
10.32ProM BPEMM plug-in time measures for all BP cases . . . . .	274
10.33ProM BPEMM plug-in time measures for each BP case and its activities . . . . .	274
10.34ProM BPEMM plug-in time measures for the activity “Assign date for Surgery” . . . . .	275
10.35Redesign options for the activity “Assign surgery date” . . . . .	276
10.36BPEMM ProM plug-in measures for the new activity . . . . .	276
B.1. BPCIP Introduction page . . . . .	303
B.2. BM3-Redesign Business Processes example . . . . .	304
B.3. VV1-Validate Business Processes example . . . . .	305
B.4. I1-Implement BPs with services example . . . . .	305
B.5. A1-Monitor BPs execution example . . . . .	306
B.6. EM4-Analyze execution measurement results example . . . . .	306
B.7. IM1-Define improvements example . . . . .	307
B.8. Event logs example . . . . .	307

B.9. Responsible for the improvement role example . . . . .	308
B.10.Evaluation phase example . . . . .	308
B.11.Introduction page in BPSOM Web Site from MINERVA . . . . .	309
B.12.BM2-Identify Business Processes example . . . . .	310
B.13.BM2-Identify Business Processes BPMN2 model example . . . . .	310
B.14.D1-Identify and categorize services example . . . . .	311
B.15.D1-Identify and categorize services SoaML diagrams . . . . .	311
B.16.I1-Implement services . . . . .	312
B.17.Work products defined in each Discipline . . . . .	312
B.18.Architect role description example . . . . .	313
B.19.Elaboration phase description example . . . . .	313
E.1. Example of users forms defined in Bonita . . . . .	389
E.2. Patient MAS BP sub-processes modeled in Bonita . . . . .	390
E.3. Example of tasks list in Bonita . . . . .	391
E.4. Example of ServiceTask as sending email in Bonita . . . . .	391
E.5. Example of queries from Bonita data base . . . . .	392
E.6. Fluxicon transformation to MXML format from Bonita execution . . . . .	392
E.7. Bonita MXML log loaded into ProM . . . . .	393
E.8. Example user forms defined in Intalio . . . . .	393
E.9. Pre-intervention sub-process modeled in Intalio designer . . . . .	394
E.10.Example tasks list showed in Intalio . . . . .	395
E.11.Web service message request in Intalio . . . . .	395
E.12.Web service message response in Intalio . . . . .	396
E.13.ProMImport framework for transforming .csv file into MXML . . . . .	396
E.14.Intalio MXML file loaded into ProM framework . . . . .	397
E.15.Resources modeling with fusion places and “chunks” for Check preconditions for MAS . . . . .	398
E.16.Resources activation page for Check preconditions for MAS . . . . .	398



# List of Tables

1.1. Summary of INGENIO project . . . . .	5
1.2. Summary of INGENIOSO project . . . . .	6
1.3. Summary of COMPETISOFT project . . . . .	6
1.4. Summary of ALTAMIRA project . . . . .	7
1.5. Summary of PEGASO/MAGO project . . . . .	7
2.1. Quantitative and qualitative research methods . . . . .	11
2.2. Threats to the validity of experiments . . . . .	17
2.3. Case study tactics for design tests from [Yin, 2002] . . . . .	19
3.1. Pros and cons of DSLs and UML profiles from [Vallecillo, 2000-2011] . . . . .	61
3.2. General structure of a typical execution event log based on [van der Aalst et al., 2007] . . . . .	70
3.3. Number of studies obtained from the selected sources . . . . .	75
3.4. Summary of primary studies selected and main principles found in each study . . . . .	85
4.1. Definition of terms in the first levels of BPMsO . . . . .	94
4.2. Definition of terms within the BPMModelElement of BPMsO . . . . .	94
4.3. Definition of terms in the first levels of SOMsO . . . . .	97
4.4. Definition of terms around Service term of SOMsO . . . . .	97
4.5. Definition of terms around Participant term of SOMsO . . . . .	98
4.6. Software Measurement Ontology (SMO) concepts from [Garcia et al., 2009] . . . . .	100
4.7. Summary of relationships defined between BPMsO and SOMsO . . . . .	104
4.8. SOMsO and SOEsO relationships . . . . .	105
4.9. BPEVsO and BPEsO and BPMesO relationships . . . . .	106
5.1. Summary of activities in Disciplines and Phases . . . . .	126
6.1. Example of execution measures specification in BPEMM . . . . .	136
6.2. Measures and Goals defined by Execution View . . . . .	136
6.3. Measures for Generic BP execution view & time dimension - Throughput Time (TT) . . . . .	141
6.4. Measures for Generic BP execution view & time dimension - Capacity . . . . .	142
6.5. Measures for Generic BP execution view & cost dimension . . . . .	143
6.6. Measures for Generic BP execution view & quality dimension - Type of ending . . . . .	143
6.7. Measures for Generic BP execution view & quality dimension - Successful branch . . . . .	144

6.8. Measures for Lean execution view & quality dimension . . . . .	145
6.9. Measures for Service execution view & time dimension - Service Response Time .	148
6.10. Measures for Service execution view & time dimension - Service Throughput . . .	148
6.11. Measures for Service execution view & time dimension - Service Capacity . . . . .	149
6.12. Measures for Service execution view & quality dimension . . . . .	149
6.13. Example event log for the Patient MAS BP . . . . .	150
6.14. Calculation of execution measures at the activity level . . . . .	152
6.15. Data from execution event logs already processed . . . . .	154
6.16. Process capacity calculation for the example Patient MAS BP . . . . .	156
6.17. Cost measures for the example BP case . . . . .	157
8.1. SoaML service generation times for BPMN2 models of different size . . . . .	200
9.1. Scales for assessing BPMS characteristics . . . . .	213
9.2. BPMN modelers key user oriented characteristics . . . . .	214
9.3. Selection of characteristics evaluated for each type of engine . . . . .	225
9.4. BP engines facilities for registering execution data for measures calculation . . . . .	227
10.1. Definition of the experiment in GQM . . . . .	236
10.2. Measures for the selected dependent variables . . . . .	238
10.3. Central hypotheses for Suitability and Understandability assessment . . . . .	239
10.4. Complementary hypotheses for Suitability and Understandability assessment . . . . .	239
10.5. Design of the experiment Part 1 - Suitability . . . . .	240
10.6. Design of the experiment Part 2 - Understandability . . . . .	241
10.7. Descriptive statistics for Suitability and Understandability . . . . .	246
10.8. Results for Suitability per presentation type and model . . . . .	247
10.9. Results for Understandability per model . . . . .	248
10.10 Percentages of agreements for education and notations knowledge level . . . . .	249
10.11 Significance levels for presentation type and Model . . . . .	250
10.12 Significance levels for education, UML, SoaML and BPMN2 . . . . .	251
10.13 Questionnaire to asses the use of BPSOM in the case study . . . . .	253
11.1. Relation Partial Objectives - Chapters in this thesis . . . . .	281
11.2. Summary of publications from this thesis . . . . .	282
11.3. Publications by Partial Objectives . . . . .	282

# Abstract

Organizations are facing several challenges nowadays, one of the most important ones being their ability to react quickly to changes either to their business process (BP) models or to the software implementing them. These changes can come from different sources: external requirements from partners or the market, or new internal requirements for the way that things are carried out by the defined BPs; they may also arise from improvement opportunities detected for the BPs defined, based on BPs execution monitoring and execution evaluation that is done by the organization, and/or its partners and customers.

The increasing complexity of both BPs models and the software implementing them, requires the changes needed or the improvements to be carefully weighed against the impact their introduction will have; they ought also to be carried out in a systematic way to assure a successful development. Two key elements are to provide these requirements: the separation of BPs definition from their implementation to minimize the impact of changes in one to the other, and a process to introduce the changes or improvements in the existing BPs and/or software implementing them.

Business Process Management (BPM) provides the means for guiding and supporting the modeling, implementation, deployment, execution and evaluation of BPs in an organization, based on the BP lifecycle. The realization of BPs by means of services provides the basis for separating their definition from the technologies implementing them and helps provide a better response to changes in either of the layers defined -definition and implementation of business processes- with minimum impact on the other.

Modeling of both BP and services is a key aspect to support this vision, helping provide traceability between elements from one area to the other, so easing the analysis of the impact of changes, among other things. Models have proven to play an important role in the software development process, one of its key uses in the context of BP realization by means of services is that of designing services at a more abstract level than with specific technologies, also promoting reuse by separating services logic from its implementation.

MINERVA: Model drIveN & sErvice oRiented framework for the continuous business process im-  
proVement & reLAted tools is the framework that has been defined in this thesis work; it takes into account all the aspects mentioned, in which the SOC and MDD paradigms are applied to BPs focusing on their continuous improvement, extending an existing BP lifecycle with explicit execution measurement and improvement activities and elements. It is made up of three dimensions:

- i) conceptual, which defines the concepts that are managed throughout the framework.
- ii) methodological, which defines a methodology for service oriented development from BPs with automatic generation of SoaML service models from BPMN2 models, along with a continuous improvement process based on execution measurement of the occurrences of BPs in the organization to carry out the improvement effort.
- iii) tools support for the whole proposal based on several existing tools we have integrated, along with new ones we have developed.

The proposals in MINERVA have been validated by means of an experiment and two case studies carried out in the context of real projects in two organizations, from which, as the main result of the applications performed, it can be concluded that MINERVA can be a useful and key guide for the continuous improvement of BPs realized by services and for the development of service oriented systems from BPs, with automatic generation of service models from BP models.





# Resumen

Las organizaciones se enfrentan en la actualidad a varios retos, siendo uno de los más importantes su capacidad para reaccionar rápidamente a los cambios ya sea en sus modelos de procesos de negocio (PN) o en el software que los implementa. Estos cambios pueden provenir de distintas fuentes: requisitos externos de socios o del mercado, o nuevos requisitos internos para la forma en que las cosas se llevan a cabo por los PNs definidos; también pueden surgir de las oportunidades de mejora detectadas para los PNs definidos, en base al monitoreo y evaluación de la ejecución de los PNs llevada a cabo por la organización, y/o sus socios y clientes.

La creciente complejidad de los modelos de PNs y del software que los implementa, requiere que los cambios o las mejoras sean sopesados cuidadosamente contra el impacto que su introducción tendrá; también deben llevarse a cabo de manera sistemática para asegurar un desarrollo exitoso. Dos elementos son clave para proveer estos requisitos: la separación de la definición de los PNs de su implementación, para minimizar el impacto de los cambios de uno en otro, y un proceso para introducir los cambios o mejoras en los PNs y/o en el software que los implementa.

La Gestión de Procesos de Negocio (Business Process Management, BPM) proporciona los medios para guiar y apoyar el modelado, implementación, despliegue, ejecución y evaluación de PNs en una organización, basado en el ciclo de vida de PNs. La realización de PNs con servicios proporciona la base para la separación de su definición de las tecnologías para implementarlos, y ayuda a proporcionar una mejor respuesta a los cambios en cualquiera de las capas definidas -definición e implementación de procesos de negocio- con un impacto mínimo sobre la otra.

El modelado de PNs y servicios es un aspecto clave para apoyar esta visión, ayudando a proveer trazabilidad entre los elementos de un área a la otra, por lo tanto facilitando el análisis del impacto de los cambios, entre otras cosas. Los modelos han demostrado jugar un papel importante en el proceso de desarrollo de software, uno de sus usos principales en el contexto de la realización de PNs con servicios es el de diseñar servicios a un nivel más abstracto que con tecnologías específicas, promoviendo la reutilización separando la lógica de los servicios de su implementación.

MINERVA: Model drIveN & sErvice oRiented framework for the continuous business process imProVement & reLAted tools es el marco que se ha definido en este trabajo de tesis, que toma en cuenta todos los aspectos mencionados, en el cual los paradigmas de Computación Orientada a Servicios (Service Oriented Computing, SOC) y Desarrollo Dirigido por Modelos (Model Driven Development, MDD) se aplican a los PNs con foco en su mejora continua, extendiendo un ciclo de vida PN existente con actividades y elementos explícitos para la medición de la ejecución y mejora de PNs. El marco se compone de tres dimensiones:

- i) conceptual, que define los conceptos que se manejan en todo el marco.
- ii) metodológica, que define una metodología para el desarrollo orientado a servicios desde PNs, con generación automática de modelos de servicio en SoaML desde modelos en BPMN2, junto con un proceso de mejora continua basado en la medición de la ejecución de las ocurrencias de los PNs en la organización para llevar a cabo el esfuerzo de mejora.
- iii) soporte de herramientas para la propuesta completa basado en la integración de varias herramientas existentes, junto con otras nuevas que hemos desarrollado.

Las propuestas de MINERVA han sido validadas por medio de un experimento y dos casos de estudio realizados en el marco de proyectos reales en dos organizaciones, de los cuales, como resultado principal de las aplicaciones realizadas, se puede concluir que MINERVA puede ser una guía útil y clave para la mejora continua de PNs realizados por servicios y para el desarrollo de sistemas orientados a servicios desde PNs, con generación automática de modelos de servicio a partir de modelos de PN.



# Chapter 1.

## Introduction

This Chapter presents an introduction to this thesis work, giving an overview on the topics it deals with. In the first place the motivation for the research work is presented in section 1.1 after which the research hypothesis and objectives are described in section 1.2, the context in which this thesis has been carried out is presented in section 1.3, and finally in section 1.4 the structure of this thesis document is described.

### 1.1. Motivation

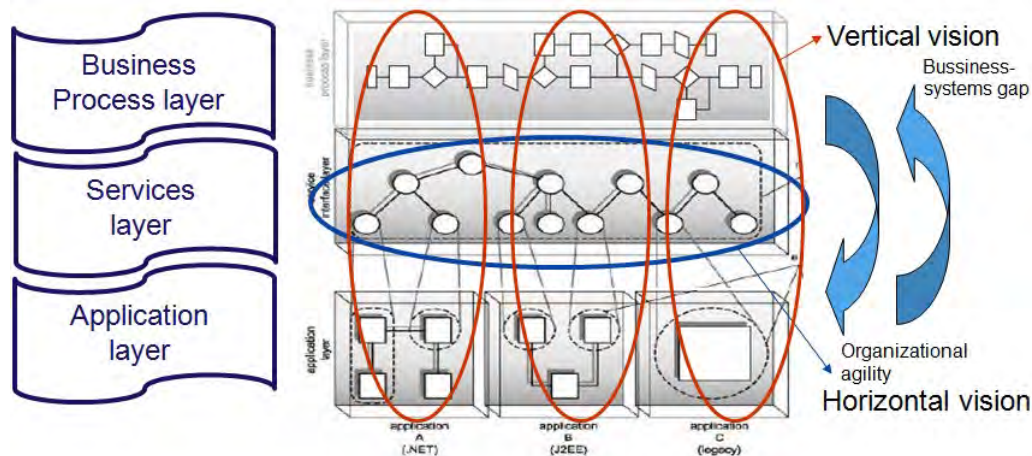
The Business Process Management (BPM) [Weske, 2007, Smith and Fingar, 2003, van der Aalst et al., 2003b] paradigm has been used by the business discipline to define, manage, optimize and improve its business processes for many years now, supporting the phases of the business process lifecycle [Weske, 2007]. In the last decade it has been embraced progressively by the software area, as business processes and their lifecycle have gained more importance for organizations that are focusing on explicitly controlling their way of doing business by means of their business processes and the software to realize and execute them. Modeling its business processes (BP) allows an organization to think about its way of conducting business, while helping it discover weakness in its processes.

Another key challenge organizations face nowadays involves their ability to react quickly to changes either to their BP models or to the software and information systems supporting them [Erl, 2005, Krafzig et al., 2005]. These changes can come from different sources: external requirements from partners or the market or from internal new requirements for the way that things are being carried out by means of the BPs defined, or improvement opportunities detected for the defined BPs based on BPs execution monitoring and execution evaluation, that is done by the organization and/or its partners and customers. In addition, the possibilities provided by Internet and globalization have raised several challenges to the way organizations usually conduct their business, as well as to the manner in which they can interact with other organizations.

A key issue refers to the integration of several already-existing independent software systems supporting different parts of the BP being implemented, which were defined in a vertical way based on different areas or sections in the organization even in different technologies, so that they can act together as a whole system. This integration has required great effort from the software area, without the expectations of the business area ever actually being fulfilled, either functionally, or in terms of budget and costs, in what is known as the business-system gap [Erl, 2005, Krafzig et al., 2005]. A horizontal vision of the organization based on business processes is needed, to provide the basis for helping close this gap and for supporting organizational agility.

The realization of business processes by means of services following the Service Oriented Computing (SOC) paradigm [Papazoglou et al., 2007] provides the basis for separating their definition from the technologies implementing them supporting the horizontal vision of the organization. Moreover, it helps provide a better response to changes in either of the layers defined -definition and implementation of business processes- with minimum impact on the other by introducing an intermediate service layer between them, which allows organizations to introduce changes and new requirements in a more agilely way [Erl, 2005, Krafzig et al., 2005]. SOC therefore allows us to change the duality process (business) - system (information) for the duality process (business) -

service (software) to tackle the continuous improvement and agile change tasks. In Figure 1.1 the different visions and realizations of business processes are shown, adapted and extended from [Erl, 2005].



**Figure 1.1.:** Vertical and horizontal visions for business processes realization based on [Erl, 2005]

On the other hand, the specification and use of models as defined by the Model Driven Development (MDD) [Mellor et al., 2003, Schmidt, 2006] paradigm allows, among other things, the explicit tracing of the relationships between elements in different models that promotes the reuse of the knowledge embedded in the transformations defined between different models and metamodels. Modeling of both BP and services is a key aspect in supporting the horizontal vision of the organization, helping provide traceability between elements from one area to the other, so easing the analysis of the impact of changes, among other things [Erl, 2005, Krafzig et al., 2005]. In the context of collaborative BP between several organizations, the ability to integrate changes, both in the definition of the BPs and the technology implementing them, with minimum impact on each other, is even more important than it is to a single organization with its internal BPs.

According to [Gartner, 2011] “.. organizations carry out BPM projects in order to improve one or more business processes ...” positioning BP improvement as the number one motivation for BPM. In the same survey the top five business goals detected include: improving customer satisfaction, improving the quality of BPs, reducing costs, improving BP agility and supporting continuous process improvement. Continuous process improvement refers to a status in which the organization is continuously analyzing the way it carries out its business, finding improvement opportunities for performing their BPs [OMG, 2008b]. The increasing complexity of both BPs models and the software implementing them, requires the changes or improvements needed to be carefully weighed against the impact their introduction will have, and to be carried out in a systematic way, to assure a successful development. Once improvement opportunities are detected, changes in the BPs that will lead to an improved new version of it must be defined, with the aim of achieving its defined business goals.

Execution measurement becomes the enabler towards understanding and controlling the real occurrences of BPs in the organization to establish a continuous BP improvement culture [OMG, 2008b]. Measurement of BPs execution provides the basis for analyzing the real behavior in the organization, helping detect deviations from the planned behavior, that will lead to finding improvement opportunities for the BPs. Business Intelligence (BI) [Castellanos et al., 2009] focuses on the collection and analysis of execution information (from BPs and other systems) to support decision making, including terms such as Business Activity Monitoring (BAM), traditional Online Analytical Processing (OLAP) and Process Mining (PM) [van der Aalst, 2011], which helps provide the business area with the needed information on real BPs execution. However, there is a general lack of an integrated view containing the complete information needed to analyze BPs execution including information about systems execution, to provide the business area with the needed operational support.

An integrated approach to carry out improvement efforts systematically, based on BPs in organizations is needed, one that can bring together all the elements mentioned, which leads to this thesis work. Its goal is the definition of a framework, which we have named MINERVA: Model driven and sERVICE oRIented framework for the continuous improVement of business process & relAted tools, to support the continuous improvement of BPs realized by services with a model-driven approach. MINERVA framework integrates the realization of BPs by means of services with a model-driven approach and methodology, providing support for their traceability and the agilely introduction of improvements; the definition, implementation, collection and calculation of execution measures, for both BPs and services execution, that are presented in an integrated view which allows business people to find improvement opportunities in BPs; and a guide and improvement activities to carry out the integration of improvements, both in BPs and services, to help achieve the business goals defined in the organization as well as the specific ones for each BP.

## 1.2. Hypothesis and objectives

The research hypothesis for this thesis work is:

*New paradigms such as Service Oriented Computing (SOC) and Model Driven Development (MDD) can offer support to set up a continuous improvement cycle in organizations for their business processes.*

Consistent with the title of the thesis, with the above hypothesis and the motivation given above, the overall objective of this thesis work is as follows:

*To define a framework to provide support for the continuous improvement of business processes based on SOC and MDD*

which is subject to the following characteristics:

- based on the integration of the paradigms: BPM, SOC and MDD.
- based on the BP lifecycle as defined by [Weske, 2007].
- considers services as the mechanism to implement BPs (SOC paradigm).
- models and metamodels are considered to be first class citizens in the framework (MDD paradigm).
- taking into account collaborative BPs between different organizations.
- integrating standard regulations and technologies (BPMN, MDA, UML, SoaML, XPDL, WS-BPEL, BPMM, etc.).
- considers BPs execution measurement as the main way to gather information about the real BP instances in the organization.
- built up of tree kinds of elements: i) conceptual (what); ii) methodological (how); and iii) instrumental (tools).

To better define the scope of this thesis work the overall objective is broken down into the following partial objectives (PO), based on the definition of characteristics as presented above:

- **PO.1:** Study the paradigms BPM, SOC and MDD and the main current proposals related to the application of SOC and MDD to BPM.
- **PO.2:** Study the main standards for languages and metamodels to: (1) model BPs and services and (2) represent the execution of collaborative BPs.
- **PO.3:** Define the overall structure and elements to be integrated in the framework.
- **PO.4:** Adapt and integrate into the framework a service-oriented methodology for service systems development from BPs.

- **PO.5:** Study and define concepts and relationships (ontology) and transformations between BP models and service models.
- **PO.6:** Define the tools support for carrying out service oriented development from BPs with a model-driven approach, and implement prototypes.
- **PO.7:** Adapt and integrate a continuous process improvement approach into the framework focusing on BPs execution measurement.
- **PO.8:** Define a set of execution measures for the assessment of BPs execution implemented by services.
- **PO.9:** Define techniques and tools for the analysis of the execution of BPs implemented by services, and implement prototypes.
- **PO.10:** Validate the proposals of the framework by means of experiments (generation of services) and case studies.

## 1.3. Context

In this section the context for the development of this thesis work is presented, in the first place research groups to which the author is related which provided the environment for the development of this thesis along with the grants awarded to carry out this thesis work, and secondly the R&D projects which have also provided partial economical support to this work.

### 1.3.1. Research Groups

This author has carried out this thesis work as a member of the Alarcos Research group<sup>1</sup> of the Department of Information Technologies and Systems, University of Castilla - La Mancha, Ciudad Real, Spain. The Alarcos Research Group was founded by professors Dr. Mario Piattini and Dr. Francisco Ruiz in September 1997, its main focus being on Software Engineering and quality of Information Systems, it is made up of forty members in all including professors and lecturers, of whom twelve have PhD.

This author is a member of the COAL Research group<sup>2</sup> of the Computer Science Institute, Faculty of Engineering, University of the Republic, Montevideo, Uruguay, and began her work at Alarcos in 2007 with a research grant from the ALFA LERnet program (Language Engineering and Rigorous Software Development)<sup>3</sup> from the European Union (February 2008 - March 2009) followed by a research grant from the Uruguayan government (Agencia Nacional de Investigación e Innovación, ANII, National Agency of Research and Innovation)<sup>4</sup> (September 2009 - February 2012). Her research interests include business processes, service oriented development, model driven development, software design and architecture and software processes and improvement.

### 1.3.2. R&D projects

The main R&D projects supporting the development of this thesis work were the INGENIO and INGENIOSO projects, although several other R&D projects, such as COMPETISOFT, ALTAMIRA and PEGASO/MAGO, with national or regional funding have also partially supported it. These are summarized in Figure 1.2 correlating each project with the Partial Objectives (PO) carried out in its context.

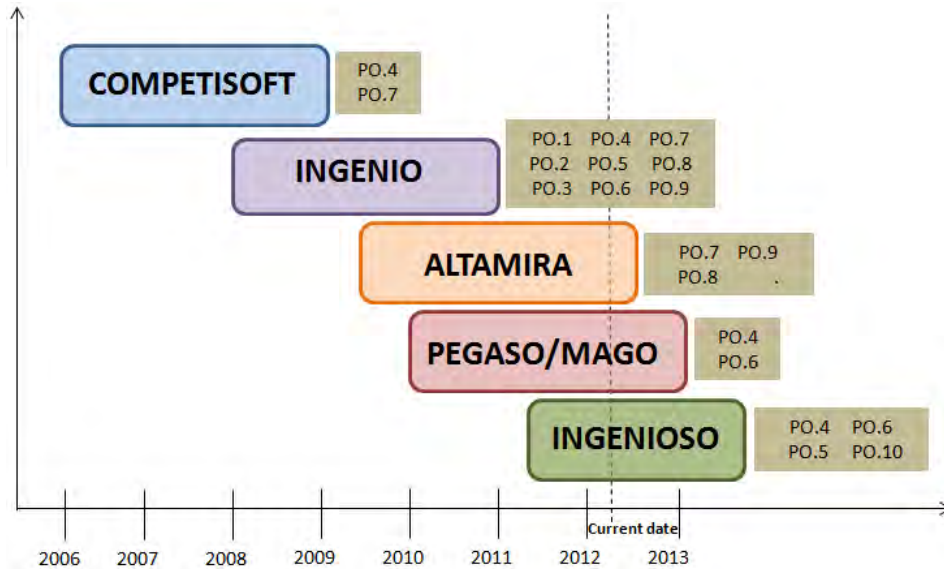
---

<sup>1</sup><http://alarcos.esi.uclm.es/defaultEng.aspx>

<sup>2</sup><http://www.fing.edu.uy/inco/grupos/coal/en/field.php/Main/Principal>

<sup>3</sup><http://alfa.di.uminho.pt/~lernnet/homepages.php?id=homepages>

<sup>4</sup><http://www.anii.org.uy/web/proyectos-beneficiarios-fortalecimiento-rrhh/BDE>



**Figure 1.2.:** R&D projects context of this thesis work

In the following, a brief summary of each R&D project is presented, focusing on the INGENIO and INGENIOSO projects as these were the fundamental projects for the development of this thesis.

### 1.3.2.1. INGENIO

The INGENIO project aims to address the improvement of business processes in an integrated manner to provide solutions of both a methodological and a technological nature. Regarding the methodology aspect, the goal is to provide organizations with the means for the effective definition of its processes, facilitating the evolution and adaptation of these models in changing business environments as well as giving the necessary support for the flexibility of its processes at runtime. Methods and mechanisms to enable organizations to carry out the measurement of its processes and related artifacts effectively and consistently will also be provided.

Regarding the technological view, the goal is to provide the necessary means for the automation of organizational/business activities and the communication between automated information systems, for which the MDD approach (Model-Driven Development) will be applied adopting a Service-Oriented approach (SOC, Service Oriented Computing). The proposals will be based on the application to the world of business of techniques that have proved useful in the world of Software Engineering. The main focus of the project is on BP modeling, measurement and automation based on SOC and MDD.

**Table 1.1.:** Summary of INGENIO project

Title	INGENIO: Aplicación de buenas prácticas de la Ingeniería del Software para la Mejora de los Procesos de Negocio (Application of Software Engineering best practices for the improvement of Business Processes)
Funding entity	Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia (Regional Government of Castilla-La Mancha, Counseling of Education and Science), Spain, PAC08-0154-9262
Grant (Euros)	130.000,00 €
Participating entities	University of Castilla-La Mancha (Spain), Technical University of Valencia (Spain)
Number of participants	18
Duration	January 2008 to December 2010
Main Researcher	Dr. Felix García

### 1.3.2.2. INGENIOSO

The business focus of the project is the Software Factories (SFs), the work will be carried out at the level of business applying an engineering point of view, known as Business Engineering. Basically, three paradigms will be applied: MDE (Model-Driven Engineering), SOC (Service-Oriented Computing) and BPM (Business Process Management). The main objective is to provide SFs with a technological framework to help them integrate the business view with the software engineering view. To do this, previous research in the following areas will be continued and reoriented: BPs, software process engineering, software processes lines, process improvement. This will be complemented with new research in the following areas: enterprise architecture in a globalized environment focused on services, model-driven business engineering, business services engineering, alignment between business and technology.

**Table 1.2.:** Summary of INGENIOSO project

Title	INGENIOSO: INGeniería de Negocio para fábricas de Software (Business Engineering for Software Factories)
Funding entity	Junta de Comunidades de Castilla-La Mancha, Viceconsejería de Ciencia y Tecnología (Regional Government of Castilla-La Mancha, Counseling of Education, Science and Culture), Spain, PEII11-0025-9533
Grant (Euros)	130.000,00 €
Participating entities	University of Castilla-La Mancha (Spain)
Number of participants	8
Duration	April 2011 to April 2013
Main Researcher	Dr. Francisco Ruiz

### 1.3.2.3. COMPETISOFT

This project aimed to provide a strategy for increasing the level of competitiveness of Latin American small and medium software organizations. This goal was achieved through the creation and dissemination of a common methodological framework for the improvement and certification of software processes, adapted to the typical characteristics of the Latin American software industry. The methodological framework is composed of: (i) a Process reference model, (ii) an Improvement framework including a continuous improvement process for guiding the improvement effort, and (iii) a Process evaluation model.

**Table 1.3.:** Summary of COMPETISOFT project

Title	COMPETISOFT: Mejora de procesos para fomentar la competitividad de la pequeña y mediana industria del Software de Iberoamérica (Process improvement to enhance the competitiveness of small and medium software industry in Latin America)
Funding entity	CYTED Programa Iberoamericano de Ciencia y Tecnología (Ibero-American Science and Technology Program), Spain, 506PI0287
Grant (U\$S)	120.00,00 U\$S (american dollars)
Participating entities	University of Castilla-La Mancha (Spain), National Autonomous University of Mexico (México), University of Cauca (Colombia), University of the Republic (Uruguay) and other 23 Universities of Latin American, more than 10 small software companies, and one national body for standardization and certification.
Number of participants	68 researches from 27 Universities and members from software companies, body of standardization.
Duration	January 2006 – December 2008
Main Researcher	Dr. Mario Piattini



#### 1.3.2.4. ALTAMIRA

The project aims to promote the improvement to high levels of maturity in software companies through the introduction of techniques related to advanced modeling, generic measurement and quantitative processes management in an environment of continuous improvement.

**Table 1.4.:** Summary of ALTAMIRA project

Title	ALTAMIRA: Aplicación de Técnicas Avanzadas de Modelado y Gestión Cuantitativa de Procesos para la Mejora de la Madurez de Fábricas de Software en niveles superiores (Application of Advanced Modeling and quantitative Management Techniques for Improvement of the Maturity of Software Factories in higher levels)
Funding entity	Junta de Comunidades de Castilla-La Mancha, Spain, Fondo Social Europeo (Regional Government of Castilla-La Mancha, Spain, European Social Fund), PII2I09-0106-2463
Grant (Euros)	240.000,00 €
Participating entities	University of Castilla-La Mancha (Spain), Indra Software Labs
Number of participants	11
Duration	April 2009 to April 2012
Main Researcher	Dr. Felix García

#### 1.3.2.5. PEGASO/MAGO

In this project the application of different techniques of software engineering processes to improve the quality of software developed in global environments will be investigated. These techniques will be contrasted experimentally using Empirical Software Engineering. The anticipated contributions fall into four areas: Definition of adaptable and flexible software processes, development of a multi-model framework for improving software quality, design of advanced techniques for quantitative management of software processes, development of a method for global requirements engineering.

**Table 1.5.:** Summary of PEGASO/MAGO project

Title	PEGASO/MAGO: Mejora Avanzada de procesos software GLObales (Advanced Improvement of Global Software processes)
Funding entity	Ministerio de Ciencia e Innovación, Spain (Ministry of Science and Innovation), TIN2009-13718-C02-01
Grant (Euros)	724.305,99 €
Participating entities	University of Castilla-La Mancha (Spain), University of Murcia (Spain) (proyecto coordinado PEGASO: Procesos para la mEjora del desarrollo GlobAl del Software, coordinated proyect PEGASO: Processes for the Improvement of Global Software development)
Number of participants	28
Duration	January 2010 to December 2012
Main Researcher	Dr. Mario Piattini

## 1.4. Document structure

This thesis document is structured in eleven chapters and several appendices, plus the bibliography references and acronyms, the contents of the rest of the chapters being as follows:

### **Chapter 2 - Research Methods:**

This chapter presents research methods and how they have been used in the context of this thesis work, describing Action-Research as qualitative method, Experimentation and Case Studies as empirical Software Engineering methods as quantitative or qualitative, along with systematic literature reviews.

### **Chapter 3 - State of the Art:**

This chapter presents the state of the art for the main topics of this thesis work: Business Process Management (BPM), Service Oriented Computing (SOC), Model Driven Development (MDD) and Continuous Process Improvement (CPI) which provides with the concepts and definitions used in this work. It also includes the results of a systematic literature review on the application of SOC and MDD to BPM which provided the author with the main current proposals at the beginning of this thesis, along with the identification of seven main principles to be taken into account when carrying out such integration.

**From Chapter 4 to Chapter 10 the main research work of this thesis is presented grouped according to the main topics defined.**

### **Chapter 4 - MINERVA framework**

This chapter details the main proposal of this thesis, the MINERVA framework. It is defined by means of two main views: dimensions and process. The Dimensions view is in turn structured using three dimensions: i) Conceptual which defines a conceptual architecture, a set of metamodels and an ontology comprising seven sub-ontologies to support the BP lifecycle; ii) Methodological which provides the approaches (methods and techniques) to be used consisting of a continuous improvement process and execution measurement model to guide the measurement effort, together with a service oriented development methodology with automatic generation of service models from BP models; and iii) Tools support which integrates existing and newly developed tools to support the work throughout MINERVA. The Process view defines the lifecycle and method of work of MINERVA, which extends the BP lifecycle [Weske, 2007] with explicit measurement and improvement activities.

### **Chapter 5 - Business Process Continuous Improvement Process (BPCIP)**

This chapter describes BPCIP which proposes an integrated approach for the continuous improvement of BPs based on extending the BP lifecycle [Weske, 2007] with explicit measurement and improvement activities, which defines the MINERVA lifecycle, focusing on BP execution measurement. The chapter presents the disciplines, activities, roles and artifacts defined by BPCIP, as well as the phases it comprises.

### **Chapter 6 - Business Process Execution Measurement Model (BPEMM)**

This chapter describes the BPEMM which integrates several BP execution measures organized in three dimensions: i) three views corresponding to Generic BP (measures that can be applied to BPs of any kind), Lean (measures to help detect non value activities in the BP) and Services (measures for the execution of services implementing the BP); ii) the “devil’s quadrant” dimensions of time, cost, quality and performance; and granularity dimension defining three levels: activity instances, BP cases and BP (all BP cases).

### **Chapter 7 - Business Process Service Oriented Methodology (BPSOM)**

This chapter describes BPSOM which proposes an extension for existing software development processes for the specific development of service oriented systems from BPs, based on the use of BPMN2 and SoaML standards for BP and service modeling respectively. The chapter presents the disciplines, activities, roles and artifacts defined by BPSOM, as well as the phases it comprises.

### **Chapter 8 - Generation of SoaML models from BPMN2 models**

This chapter describes the automatic generation of SoaML service models from BPMN2 models by means of the definition of QVT transformations based on correspondences first identified between service and BP models concepts (ontology relationships), and then transferred to correspondences between elements of the BPMN2 and SoaML metamodels.

### **Chapter 9 - Tool support**

This chapter describes the set of existing tools that have been integrated into MINERVA along with the new ones developed in the context of this thesis, to support all the activities and the generation of artifacts as defined in the MINERVA method of work.

### **Chapter 10 - MINERVA framework validation**

This chapter presents the validation of the MINERVA framework carried out by means of an experiment and two case studies. The experiment was defined to assess the suitability of the QVT transformations defined (i.e. from BPMN2 models to SoaML models) as well as the understandability of the results of the QVT transformations (i.e. the SoaML models). One case study corresponds to the validation of BPCIP and BPEMM in the context of a project with the Ciudad Real General Hospital (HGCR, General Hospital from Ciudad Real), and the other case study corresponds to the validation of BPSOM and the automatic generation of SoaML service models from BPMN2 models (i.e. QVT transformations) in the context of the public telecommunications enterprise from the Uruguayan government, ANTEL.

### **Chapter 11 - Conclusions and future work**

This chapter presents the conclusions of this thesis work, analyzing the attainment of objectives and the contributions of the work, the scientific publications achieved, along with the research lines open for future work.

### **Appendices:**

The Appendices included extend and clarify information to give a better understanding of some of the issues presented in previous chapters. The list of Appendices is as follows:

**Appendix A** - Data extraction from primary studies of the systematic review

**Appendix B** - BPCIP and BPSOM Web Sites (implemented with EPF Composer)

**Appendix C** - QVT transformations code

**Appendix D** - Experimental material of the QVT transformations validation experiment

**Appendix E** - HGCR case study implementation in XPDL and WS-BPEL and simulation of resources



## Chapter 2.

### Research methods

This Chapter describes research methods in Software Engineering, providing concepts and definitions to give a general view on the subject, and then describing their use in this thesis. Unlike efforts on projects and other work of an operational nature, in research the method is an essential aspect to get results and to ensure their quality and validity.

The Chapter is organized as follows: in section 2.1 key concepts and definitions about research methods in Software Engineering are provided, in section 2.2 the use of research methods in this thesis is presented and finally in section 2.3 conclusions for the chapter are discussed.

#### 2.1. Research methods in Software Engineering

Understanding a discipline implies learning, that is, observation, reflection and encapsulation of knowledge, construction of models (of the application domain, of processes to solve problems), experimentation and evolution of models within time [Basili, 2000]. This view has been applied in different disciplines such as medicine, physics, and the manufacturing industry, the only differences between them depending on how the models are constructed and analyzed, and how experimentation is carried out. Software Engineering and Information Systems areas can be seen from a scientific point of view as being of experimental character, where the focus of investigation is to know the nature of processes, products and their relationships, in the context of a software or organizational system.

Different methods of research, which are mainly quantitative and qualitative methods [Myers, 1997] have been applied over recent years, these are summarized in Table 2.1. The main differences between qualitative and quantitative research have to do with the nature of the data and the data analysis. In qualitative research data is unstructured, obtained by means of interviews, observations and group discussion and it is not analyzed by means of statistical analysis, while in quantitative research data is more structured and it is analyzed by means of statistical techniques.

**Table 2.1.:** Quantitative and qualitative research methods

Aspect	Qualitative research	Quantitative research
Purpose	to provide knowledge about an organization and/or problem and its solutions	to generalize results from a case study to the total population of interest
Orientation	to verification	to discovering
Sample size	small	large
Data collection	unstructured data obtained by means of interviews, observations and group discussion	data and techniques very structured
Data analysis	non statistical	statistical

Deductive and empirical methods could be classified as quantitative research methods and are especially suited to the study of natural phenomena or objects. The study of cultural and social

phenomena, however, requires another kind of method, not based on formal experiments or theories but on interviews, questionnaires, documents, impressions and the reactions of the researcher, etc. These are called qualitative methods and they include action-research, case studies, ethnography, etc.

Empirical research can be classified into experiments, case studies and surveys [Robson, 2002]. Experiments, or controlled experiments, are characterized by “measuring the effects of manipulating one variable on another variable” [Robson, 2002], while a case study investigates a phenomenon within its real life context, when the boundaries between phenomenon and context are not really evident [Yin, 2002], and surveys for their part, are a collection of standardized information from a specific population, or sample, usually by means of a questionnaire or interview [Robson, 2002].

As stated in [Yin, 2002] “case studies can be based on any mix of quantitative and qualitative evidence... Note that as analogous examples, some experiments (such as studies of psychophysical perceptions) and some survey questions (such as those seeking categorical rather than numerical responses) rely on qualitative and not quantitative evidence.”

In this thesis Action-Research, experiments and case studies are the research methods used, along with systematic review, and will be explained in the next sections.

### 2.1.1. Action-Research

Among the various qualitative research methods proposed in the literature (mostly from the social sciences field), the one used most in Software Engineering and Information Systems is Action - Research. The term was coined by the author Kurt Lewin (1947) with which he described a way of research that could link the experimental focus of social sciences with programs of social action to respond to main social problems of that particular time. By means of action-research, Lewin stated that advances in theory and social changes could be performed simultaneously. In recent years, this method has been widely accepted and applied in software engineering research since its introduction by Wood-Harper in 1985.

Several definitions of action-research exist among which the most significative are: a way used by groups of persons to prepare the conditions necessary from them to learn from their own experiences and to make these experiences accessible to others [McTaggart, 1991]; the process of systematic gathering of research data of a current system in relation to some objective, goal or need of that system, to feed that data back to the system to take actions by means of selected alternative variables within the system, based both on data and hypothesis, and evaluating the results of actions collecting additional data [French and Bell, 1996], or the participation of all parts involved in the research, examining the existing situation (perceived as problematic) with the objective of changing and improving it [Wadsworth, 1998].

From these definitions it can be stated that action-research has a dual purpose: to generate a benefit to the “client” of the research and at the same time, to produce relevant research knowledge [Kock and Lau, 2001], that is, a collaborative way of research that seeks to unify theory and practice between researchers and practitioners by means of a cyclic process. Action-research is oriented to the production of new knowledge that is useful in practice, obtained by means of changing and/or seeking solutions to real situations that happen to a practitioners group [Avison et al., 1999]. This is achieved by means of the intervention of a researcher in the reality of the group mentioned, and results of the experience must be beneficial to both the researcher and the practitioners. A fundamental premise in this way of research is that complex social processes (and the use of information technologies in such organizations) can be studied better by introducing changes in these processes and observing the effects of these changes [Baskerville, 1999].

In the Information Systems field, the client of the research is generally an organization in which the researcher provides services such as consulting, help in changing or developing software, in exchange for having access to interesting data for the research, and sometimes receiving funding [Kock and Lau, 2001]. Anyway, the researcher that uses action-research in Information Systems (AR-IS) serves two different entities: the client of the research and the Information Systems scientific community.

Needs from both groups are in general different and sometimes opposed, so the main challenge that an AR-IS researcher has to face is to try to satisfy both demands. Four main roles, which can be played by the same person or team, are identified in this method [Wadsworth, 1998]:

- **researcher:** person or group of persons carrying out the research process.
- **researched object:** that is, the problem to be solved.
- **critical reference group:** the group that is researched in order to solve the problem, which also participates in the research process but less active than the researcher.
- **beneficiary (stakeholder):** of the research result but without direct participation in the research process.

There are four variants for action-research as proposed by [French and Bell, 1996], and these are mainly dependent on the characteristics of the research:

- **Diagnosis:** the researcher looks inside a problematic situation, diagnosing the problem and making recommendations to the critical reference group, but without tracking the effects.
- **Participative:** the critical reference group applies the researcher's recommendations, sharing their results and effects.
- **Empirical:** the critical reference group registers the effects and actions in a systematic way, which makes this option difficult to apply.
- **Experimental:** different options for achieving an objective are assessed, the main problem being the difficulty of measuring the different options objectively. This is due to the fact that in general they will be applied in different organizations with different characteristics, or in the same organization but at different times with different environments.

A research process based on Action-Research is composed of groups of activities defining a cycle, in which four main steps are identified [Padak and Padak, 1994], which are shown below.

1. **Planning:** to identify the relevant issues that will guide the research; these must be directly connected with the researched object and it must be likely that answers for them will be found.
2. **Action:** careful, deliberate and controlled variation of the practice, where a simulation of proof of the solution is carried out. This is when the researcher acts on the reality.
3. **Observation:** to collect information, data, documenting what happens; this information can come from any source (bibliography, measures, proof results, observations, interviews, documents, etc.). It is also known as evaluation.
4. **Reflection:** to share and analyze the results with the other parties involved, in a way that other relevant issues can be identified.

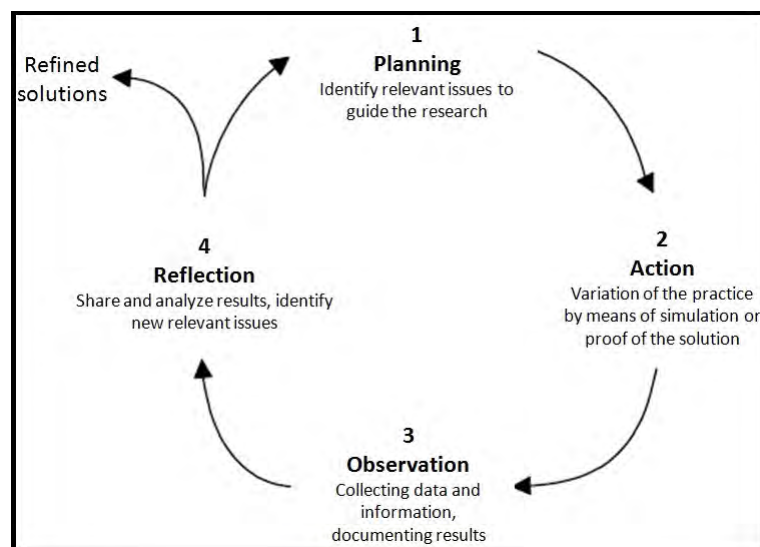


Figure 2.1.: Action-research cyclic nature

As it can be observed, the action-research process is iterative, in a way in which solutions are refined in every cycle, new ideas are proposed, applied and assessed in the next cycle, making it a reflexive process of learning and search for solutions.

## 2.1.2. Empirical Software Engineering

In this section experiments and case studies are presented as the types of empirical Software Engineering used in this thesis.

### 2.1.2.1. Experiments

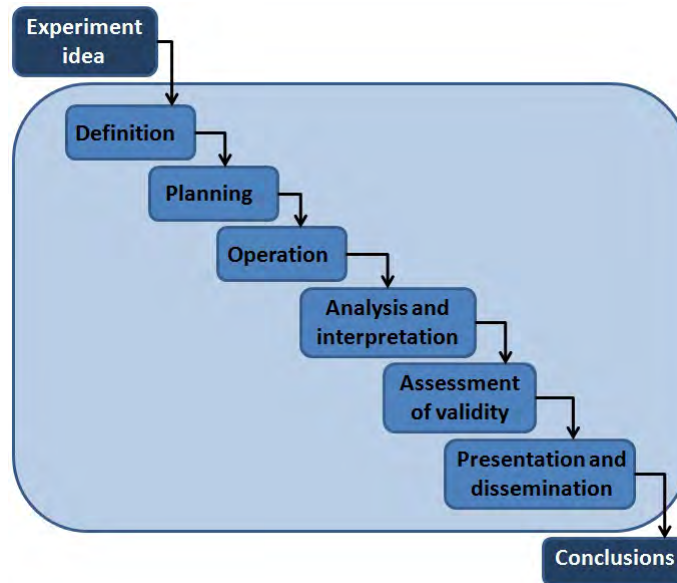
Experimentation provides a systematic, disciplined, quantifiable and controlled way of evaluating human developed activities [Wohlin et al., 2000]. An experiment is a formal, rigorous and controlled exam in which key factors are identified and manipulated, and are used when the situation is under control and their behavior can be controlled in a direct, precise and systematic way. Experiments can be carried out in a laboratory under controlled conditions, where events are organized to simulate their appearance in the real world. They can also be performed on line, that is, the research is carried out in the work field under normal conditions [Babbie, 1990], but this type of situations is the most difficult to control as it is not possible to control all factors. The objective is to manipulate one or more variables while maintaining the rest of the variables at fixed levels.

The advantage of an experiment is that it can determine in which situations given statements are true and can provide the context in which certain standards, methods and tools are recommended. Only if the experiment is carried out properly is it possible to draw conclusions about the relationships between cause and effect for which the hypothesis is formulated [Wohlin et al., 2000]. Experiments need to be planned carefully if useful and significant results are to be obtained [Juristo and Moreno, 2001].

Although experimentation in software engineering is necessary it is also very difficult [Basili et al., 1999] as there are several context variables, so to be able to understand experiment results adequately, a mechanism to explain studies and integrate results is needed. It is also important to carry out replications of the experiments, as with the results of a single experiment it is difficult to appreciate if the results can be generalized and to conclude that the results are valid. A fundamental strategy for performing replications is the creation of “laboratory packages” [Basili et al., 1999], which collect information from all the experimental material, i.e. the experimental design, artifacts, processes used in the experiment, the methods used for experimental analysis and decisions taken.

The process for carrying out experiments is composed of six main stages [Wohlin et al., 2000]: definition, planning, operation, analysis and interpretation, evaluation of the validity, and presentation and dissemination, which are shown in diagram form in Figure 2.2. This process is defined for experiments but can also be used for any empirical study.





**Figure 2.2.:** Stages in carrying out experiments from [Wohlin et al., 2000]

### Definition

In this phase the basis for the experiment is determined -the “why” of the experiment. The aim is to define the objectives of an experiment which has been formulated from a problem to be solved. Following [Briand et al., 2002, Lott and Rombach, 1996] a Goal, Question, Metric (GQM) template is used to define the objective of an experiment, which includes the following elements:

- **the object of the study**, which can be a product, a process, a resource, a model, a measure or a theory, which is the entity that is studied in the experiment.
- **the purpose**, which defines the objective or goal of the experiment.
- **the quality focus**, which is the primary effect that is studied in the experiment.
- **the perspective**, which is used to define the point of view from which the results of the experiment will be interpreted.
- **the context**, which defines the environment in which the experiment will be carried out, composed of the subjects and the artifacts used in the experiment.

### Planning

This phase establishes how the experiment will be carried out, and is in turn divided into six steps: context selection, hypothesis formulation, variables selection, subjects selection, design of the experiment and instrumentation.

- **Context selection:** the context of the experiment is characterized by four dimensions: off-line vs. on-line, students vs. professionals, simulation vs. real problems and specific vs. general (generalization of results).
- **Hypothesis formulation:** the definition of the experiment is formalized by means of hypotheses where two must be defined: the null hypothesis (H0) and the alternative hypothesis (H1), so data can be tested against the hypothesis in order to decide whether the null hypothesis can be rejected.
- **Variables selection:** the dependent variables (variables to be studied) and independent variables (variables whose values are changed to observe the effects) that will be considered have to be selected, along with the way in which they will be measured and their measure scales.
- **Subject selection:** the selection of subjects for the experiment is related to the generalization of results to a population, for which the sample must be representative of that population.

- **Design of the experiment:** an experiment consists of a series of tests of the treatment, which must be planned and designed carefully. The design describes how the tests are organized and how they will be executed. When one independent variable is used the design is simple and can be inter-subject (one treatment per subject) or intra-subject (all treatments per subject). When two or more independent variables are used the design is factorial and can be of several types such as complete (combining the factors) and partially fractioned (nesting factors).
- **Instrumentation:** it can be of three types: particular objects, instructions and measurement instruments. The objects of the experiment can be for example specification or code of documents. Instructions are needed for the participants to know what to do within the experiment, and also training in the methods to be used is needed. Measurement of the experiment is carried out within the data collected, which can be in forms or interviews. The main goal of instrumentation is to provide the means for carrying out and monitoring the experiment, which must not affect the results of the experiment.

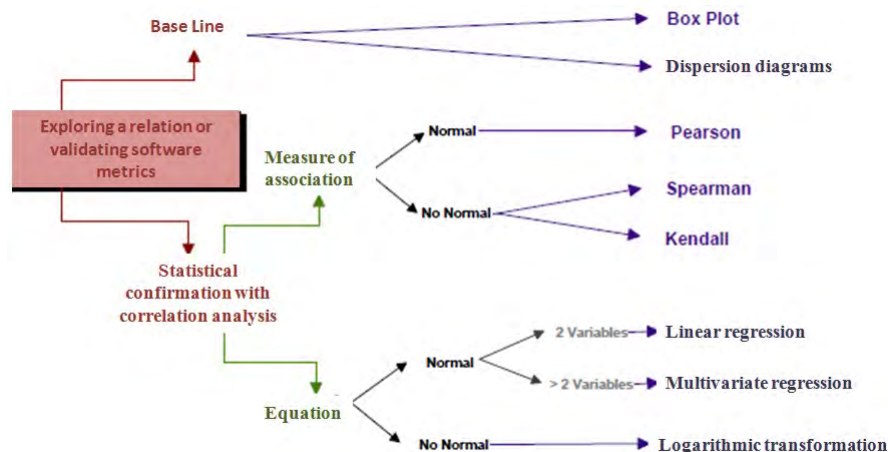
### Operation

The operation of the experiment is to actually carry out the experiment in order to collect the data to be analyzed, where the experimenter meets the subjects. It is divided into three stages:

- **Preparation:** the preparation is performed in order to find the subjects that can commit to participating in the experiment.
- **Execution:** the execution is when the experiment is carried out.
- **Data validation:** after the experiment is carried out, the experimenter must check if the data has been correctly collected and if it is reasonable.

### Analysis and interpretation

Once the data is collected it must be properly analyzed; for doing so three main elements have to be considered when choosing analysis techniques: the nature of the data collected, the reason for the experiment as well as the type of experimental design. Different techniques can be used to test the hypotheses, where the objective is to assess whether the null hypothesis can be rejected based on a sample of a statistical distribution. [Pfleger, 1994] carried out a detailed study on the several statistical tests that can be applied according to the objectives to be satisfied; this is shown in the form of a decision tree to make it easier to choose the statistical test to apply in each case.



**Figure 2.3.:** Decision tree for analysis techniques from (Pfleger, 1994)

In the experiment carried out in this thesis work, described in chapter 10, the tool SPSS (Statistical Package For Social Science) version 17 was used to apply the different statistical methods to the collected data automatically.

### Assessment of validity

A key issue related to the experimental results is the validity of these results. The degree of credibility of an experiment depends on the validity of the conclusions obtained. It is important to consider the validity at the beginning of the planning phase in order to plan the validity of the experiments properly. In Table 2.2 a list of different threats to the validity of experiments [Cook and Campbell, 1979] is presented, which have to be controlled in order for the results in any empirical study to be valid.

**Table 2.2.:** Threats to the validity of experiments

Threats to validity	Description
Conclusions validity	Low statistical power, violating the assumptions of the statistical tests, completion and error rate, reliability of metrics, reliability or implementation treatment, random irrelevancies in the experimental environment
Construct validity	Inadequate pre-operational interpretation of constructors, monooperation bias, mono-method bias, confusion between constructors and constructor levels, interaction of different treatments, testing and treatment interaction, restricted generalization through constructors, guessing hypothesis, apprehension in evaluation, experimenter's expectations
Internal validity	History, Maturity, experimentation, instrumentation, statistical regression, selection, mortality, ambiguity about the meaning of causal influence, interaction with the selection, Diffusion of duplicate treatment, compensatory equalization of treatments, compensatory rivalry, resentful demoralization.
External validity	Interaction of selection and treatment, environmental and treatment interaction, history and treatment interaction.

### Presentation and dissemination

Once the experiment has been carried out, the results are to be presented, in many cases. This could be done in an article for a conference, a report for decision making, or as educational material. For the presentation and dissemination of an experiment it is essential not to forget the important aspects and the information needed to carry out the replicas and obtain benefits from the experiment and from the knowledge gained through it.

#### 2.1.2.2. Case study

Case studies are used to monitor projects, activities or assignments, where the data is collected for a specific purpose of the study. The case study is oriented in general to analyzing a particular attribute or establishing relationships between different attributes. Several guides for carrying out and reporting case studies are proposed such as [Yin, 2002, Runeson and Höst, 2008] along with a template for reporting [Brereton et al., 2008]. The level of control in a case study is lower than in an experiment, since case studies are observational studies (i.e. carried out by the observation of an ongoing project or activity) [Zelkowitz and Wallace, 1998] while experiments are controlled studies. Most of the issues considered in carrying out an experiment are also taken into account when conducting a case study, as it requires the same steps as experiments do. The establishment of hypotheses is particularly important because they provide the guide for measuring and analyzing the results.

[Yin, 2002] presents at least five applications for case studies: the first is to explain the presumed causal links between real-life interventions that are too complex for surveys or experiments; they may also be used to describe an intervention and the real-life context in which it occurred, or they can illustrate certain topics within an evaluation in a descriptive mode; they can also be used to explore situations in which the intervention being evaluated has no clear single set of outcomes,

and finally, the case study can be a meta-evaluation i.e. a study of an evaluation study. [Yin, 2002] proposes several stages for the development of case studies, which are shown in Figure 2.4.

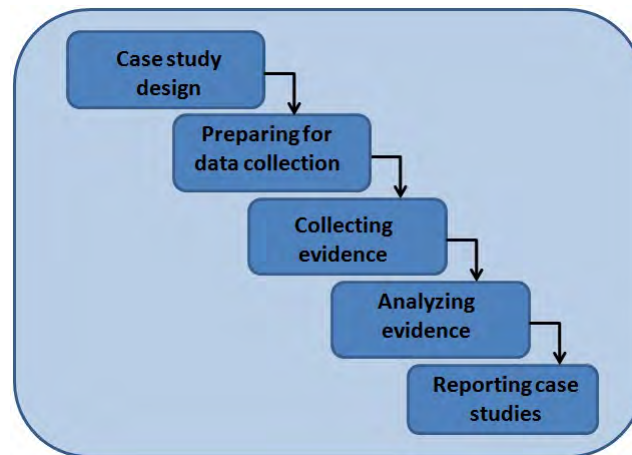


Figure 2.4.: Stages in case study development from [Yin, 2002]

### Case study design

According to [Yin, 2002] every type of empirical research has a research design (implicit or explicit), which describes the logical sequence that connects the empirical data to the initial research question and to its conclusions. It can be seen as the logical plan to getting “from here to there” where “here” represents the initial questions to be answered and “there” represents the conclusions i.e. the answers; in the middle a number of major steps may be performed. Five main components are proposed for the design of case studies:

- **questions of the study:** a basic categorization of questions is the known “who?”, “what?”, “where?”, “how?” and “why?” series. “What?” questions help to identify hypotheses and propositions of the research, “how?” and “why?” help explain operational links needed to be traced over time (cause-effect relationships), and “who?”, “where?” and others such as “how many?” or “how much?” help to describe the incidence or prevalence of a phenomenon or when it is to be predictive about certain outcomes (surveys are likely to be favorable for these ends).
- **study propositions (if any):** propositions help identify what should be studied, based on the questions.
- **unit(s) of analysis:** consist in defining “what the case is” and is/are related to the way the initial research questions have been defined. Delimiting the unit of analysis (which persons-groups-entities are inside the unit of analysis and which are outside) will determine the boundaries of the data collection and analysis.
- **the logic linking the data with the propositions:** how to connect collected data to the stated propositions; an example is “pattern matching” where patterns are defined and data is “matched” to them.
- **the criteria for interpreting the findings:** how to interpret the results by means of decision criteria to analyze them.

[Yin, 2002] also provides criteria for judging the quality of the case study design, by means of the four tests that are most commonly used: construct validity, internal validity, external validity and reliability, as summarized in Table 2.3.

**Table 2.3.:** Case study tactics for design tests from [Yin, 2002]

Tests	Case study tactics	Phase of research in which tactic occurs
Construct validity	use multiple sources of evidence, establish chain of evidence, have key informants to review draft case study report	data collection, composition (last)
Internal validity	do pattern-matching, do explanation-building, address rival explanations, use logic models	data analysis
External validity	use theory in single-case studies, use replication logic in multiple-case studies	research design
Reliability	use case study protocol, develop case study database	data collection

Four types of case design are also defined in [Yin, 2002]: firstly holistic single case studies, if the case examined only the global nature of an organization or of a program, secondly embedded single case studies where multiple units of analysis are studied within a case (sub-units), thirdly and fourthly multiple case studies when the study as a whole covers several units and each unit is the object of a case study (holistic and embedded respectively), for example a study on school innovations where each school applies innovations and several schools are covered by the case study. One advantage is that the overall study is regarded as being more robust.

### Preparing for data collection

The preparation for a case study includes [Yin, 2002]: skills for the investigator, the training and preparation for the specific case study, the development of the case study protocol, the screening of candidate case studies and conducting a pilot case study.

- **skills for the investigator:** several are defined such as that the researcher should be a good “listener”, know how to “ask good questions”, be adaptable, flexible and unbiased by preconceived notions.
- **training and preparation:** every researcher in the case study must be able to operate as a senior researcher; the training begins with the definition of research questions and case design, so the theory, methodology, and case study are well known. If other researchers have to be trained to perform the case study one option is by way of a seminar with the senior researcher, so all researchers are qualified to collect data and act in the case study.
- **case study protocol:** contains the instruments, the procedures and general rules to be followed in using the protocol. It is intended to guide the data collection from a single case study (even if it is part of a multi-case design) so it is a way of increasing reliability. As a general guide it should include:
  - an overview of the case study project: project objectives and auspices, case study issues, and relevant readings about the topic being investigated.
  - field procedures: access to case study sites, credentials, general sources of information, and procedural reminders.
  - case study questions: specific questions that the researcher must keep in mind when collecting data and potential sources for answering questions.
  - a guide for the case study report: outline, format for the data, use and presentation of other documents and bibliography.
- **screening candidate case studies:** the final selection of sites or individuals that will serve as the case study, identifying the case study from several possibilities, and sometimes to carry out small case studies to be able to select the most appropriate.

- **conducting a pilot case study:** this can be seen as a “laboratory” for the researchers to improve the questions, the protocol for data collection or even the case study design.

### Collecting evidence

Data for case studies come from six main sources: documentation (including minutes of meetings, written reports or events, progress reports, other evaluations of the site, etc.); archival records in the organization (such as service records: i.e. number of clients per period of time, organizational records: charts and budgets, survey data: previously collected in the organization); interviews (one of the most important sources in case studies); direct observation (visit the case study site), participant-observation (assuming a participating role in the case study) and physical artifacts (technological device, tool or instrument). Three main principles for data collection are also defined in [Yin, 2002]:

- Use multiple source of evidence: using data from more than one source, and defining converging lines of inquiry by means of triangulation which can be of four types: of data sources (data triangulation), among different evaluators (investigator triangulation), of perspectives to the same data set (theory triangulation) and of methods (methodological triangulation).
- Create a case study database: to store the data collected making this data available to new researchers and for inspection, including answers to questionnaires and interviews.
- Maintain a chain of evidence: between initial research questions, data collected and conclusions, which allows an external observer to follow the derivation of any evidence ranging from the initial research questions to the conclusions, allowing the steps in any direction to be traced (from conclusions back to initial research questions or viceversa).

### Analyzing evidence

As stated in [Yin, 2002] data analysis consists of examining, categorizing, tabulating, testing, or recombining both quantitative and qualitative evidence to address the initial propositions of the case study. The best preparation for conducting case study analysis is to have a general analytic strategy, allowing the data to be reasonably interpreted, for which three strategies are proposed:

- relying on theoretical propositions: the most preferred strategy is to follow the theoretical propositions that led to the case study, where propositions help to focus attention on certain data and to ignore other data.
- thinking about rival explanations: to define and test rival explanations, where the more rivals that can be defined and rejected the more confidence can be placed on the findings.
- developing a case description: to develop a descriptive framework for organizing the case study, when the two strategies presented previously are difficult to achieve.

Several analytical techniques which will help in dealing with internal and external validity in doing case studies can be used within all strategies presented, such as pattern matching, explanation building, time-series analysis, logic models, etc.

### Reporting case studies

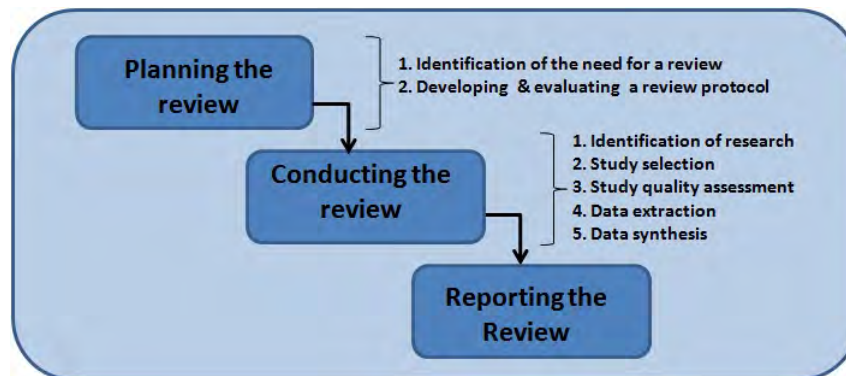
Implies bringing its results and findings to closure, and regardless of the form of the report several guidelines have to be taken into account, such as the audience of the report, the compositional structure of the report, following certain procedures such as having the report reviewed by persons who are the subject of the case study. An exemplary case study must be: significant, “complete”, must consider alternative perspectives, must display sufficient evidence, must be composed in an engaging manner (to attract the reader).

### 2.1.3. Systematic review

A systematic review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest [Kitchenham, 2004, Kitchenham and Charters, 2007]. Systematic reviews aim to present a fair evaluation of a research topic by

using a trustworthy, rigorous, and auditable methodology. Although most research works starts with a literature review of some sort, unless this is thorough and fair, it will be of little scientific value, and this is the reason for undertaking systematic reviews [Kitchenham, 2004, Kitchenham and Charters, 2007].

The method proposed by Kitchenham [Kitchenham, 2004, Kitchenham and Charters, 2007] defines a systematic review consisting of three stages which are shown in Figure 2.5: planning the review, development of the review and publication of the results.



**Figure 2.5.:** Three stages for systematic reviews defined by [Kitchenham and Charters, 2007]

Although the three stages defined may appear to be sequential, many of them involve iteration, in particular when the development of the review protocol is started many activities are initiated and refined afterwards, the inclusion and exclusion criteria defined, for instance, can be refined after quality criteria is defined, or data extraction forms produced initially can be amended when quality criteria is defined, or data synthesis methods defined in the protocol can also be amended once the data has been collected [Kitchenham, 2004].

The stages defined include several elements, starting from the definition of the research question, key words and research chains, the execution of sources for the chains defined, and the inclusion and exclusion criteria for choosing relevant and primary studies from which to extract the associated data, which are presented in the following.

### Planning the review

The importance of planning the review, as in experiments or case studies, is to explicitly define the steps and work to be done, and the main elements for guiding that work, by means of:

- identification of the need for the review: which mostly responds to requirements of researchers to summarize existing information about some phenomenon, with the objective of drawing more general conclusions on it or as a prelude to further research activities.
- development of a review protocol: the review protocol specifies the methods that will be used to undertake a specific systematic review, and is necessary to reduce the possibility of researcher bias. It must define several things:
  - background: defines the rationale for the survey.
  - the research questions that the review intends to answer: a guideline is included showing different types of questions for software engineering such as, among others: assessing the effect of a software engineering technology, the frequency or rate of project success or failure, identifying cost and risk factors associated with a technology. When the systematic review is the basis for future work the questions will identify and/or scope these future research activities.
  - the strategy that will be used to search for primary studies: including search terms, search strings, and resources to be searched including databases, specific journals, and conference proceedings.

- study selection criteria and procedures: determining criteria for including a study in, or excluding it from, the systematic review. Studies fulfilling the inclusion criteria are the relevant studies, from which the ones fulfilling the exclusion criteria are rejected providing the primary studies which contain key information regarding the research question of the review.
- study quality assessment checklists and procedures: the researchers should develop quality checklists to assess the individual studies.
- Data extraction strategy: to define how the information required from each primary study would be obtained.
- synthesis of the extracted data: to define the synthesis strategy, clarifying whether or not a formal meta-analysis is intended and if so what techniques will be used.
- project timetable: to define the review plan.

### **Development of the review**

The search strings defined by the protocol, are executed among the selected sources, adapting them to each of the search engines. This is done as to be able to obtain as many studies as possible, from which the ones which do not fulfill the inclusion criteria are rejected, obtaining the relevant studies. After ruling out repeated studies the non-repeated studies left are read in depth and if the exclusion criteria are not fulfilled they are selected as primary studies, if, on the other hand, the exclusion criteria are fulfilled the studies are rejected. From the primary studies, the data extraction is carried out to analyze the results. The steps defined are as follows:

- identification of research: the objective is to find as many primary studies as possible so an unbiased search strategy is needed avoiding, for example, language bias or publication bias (selected sources).
  - A search strategy should be defined based on searches in selected sources using the search strings, which are usually iterative and include performing several trial searches using a combination of the terms derived from the research question and the logical operators “AND”, “OR”, etc., assessing the results.
- selection of studies: the criteria for study selection are meant to identify the primary studies that provide direct evidence about the research question, and must be defined during the protocol definition, based on the research question (both inclusion and exclusion). They should be piloted to ensure that they can be reliably interpreted and that they classify studies correctly.
- study quality assessment: it is important to assess the quality of primary studies to be able to provide information such as if quality differences provide an explanation for differences in study results or to weight the importance of individual studies in the synthesis of studies. Detailed quality assessments are usually based on a checklist of factors to be assessed for each study.
- data extraction and monitoring progress: involves defining data extraction forms to collect all the information needed to address the review questions and the study quality criteria, including, among other data, information such as title, authors, journal, publication details. When possible, data extraction should be performed independently by two or more researchers.
- data synthesis: involves collating and summarizing the results of the primary studies included, by means of two kind of synthesis:
  - descriptive synthesis: information extracted about the studies (i.e. intervention, population, context, sample sizes, outcomes, study quality) should be tabulated in a manner consistent with the review question
  - quantitative synthesis: to synthesize quantitative results from different studies, study outcomes must be presented in a comparable way, using analytic techniques to obtain means, odds, etc.



### Publication of the results

It is important to communicate the results of a systematic review effectively. Usually systematic reviews will be reported in at least two formats: in a technical report or in a section of a PhD thesis and in a journal or conference paper. The systematic review performed in this thesis work has been published in a conference and extended in the CCIS Springer series corresponding to selected papers from the conference, as is described in chapter 11.

## 2.2. Use of research methods in this thesis

In this section the use of the research methods presented in this thesis is described: a) Action-Research for the definition of the framework and tool support; b) an experiment to validate the QVT transformations to generate SoaML service models from BPMN2 models; c) case studies to validate the framework proposal and d) a systematic review to find existing proposals on the application of SOC and MDD to BPM at the beginning of this thesis.

### 2.2.1. Action-Research

As the main objective of this thesis is “to define a framework to provide support for the continuous improvement of business processes based on SOC and MDD”, for the definition of the framework the participative variant of Action-Research was chosen defining the following participants:

- **researcher:** Alarcos Research Group, which is made up of teaching staff from the Faculty of Computer Science from the University of Castilla - La Mancha, along with the author of this thesis work.
- **researched object:** in this thesis it is the “framework to provide support for the continuous improvement of business processes based on SOC and MDD”.
- **critical reference group:** Hospital General de Ciudad Real (HGCR, General Hospital of Ciudad Real) specifically the Quality Group that is in charge of the BPs that have been modeled in a previous work, and the Information Technology (IT) group.
- **beneficiaries (stakeholders):** organizations that want to integrate BPM and execution measurement to improve their BPs with SOC and MDD; in this specific case these are the Hospital General de Ciudad Real (HGCR), as well as the software development organizations that want to improve the development of BPs by means of SOC and MDD.

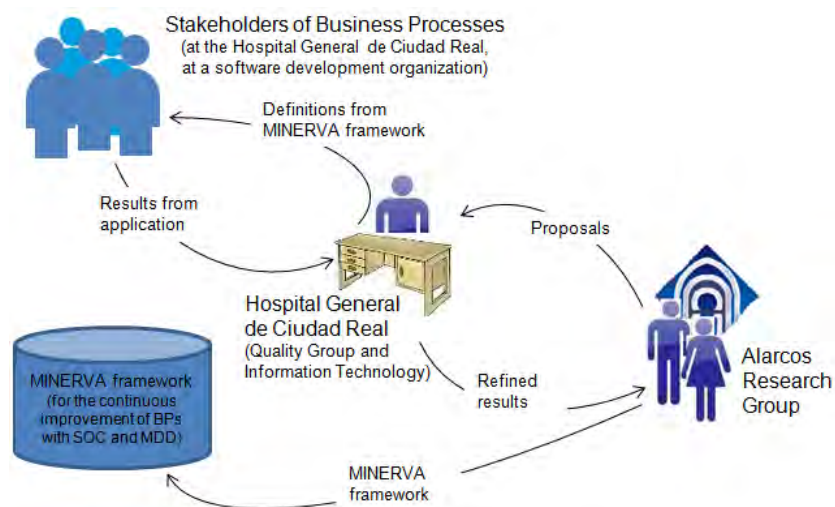


Figure 2.6.: Application of Action-Research in this thesis work

The most evident application of action-research in Information Systems (AR-IS) is when a human organization interacts with information systems. In fact, action-research is one of the few valid approaches for studying the effects of specific alterations in methodologies for development and maintenance of systems in human organizations [Baskerville and Wood-Harper, 1996]. This means that the definition of a framework for the continuous improvement of BPs implemented by services with a model-driven approach is a suitable domain for the application of action-research.

The general approach consisted in: 1) the researcher proposed a theoretical framework accepted by the critical reference group, 2) the researcher worked actively, so the benefits were mutual, scientific benefits for the researcher and practical benefits for the critical reference group, 3) the knowledge obtained was applied in the organization immediately (when possible) and 4) the research was developed in a typical cyclic and iterative process combining theory and practice. These solutions were materialized in components of the MINERVA framework that were proposed and analyzed, based on an initial cycle followed by general cycles as summarized in the following:

- Initial cycle: the problematic was defined as being related to the application of SOC and MDD to BPM and the improvement of BPs based on execution measurement and the BP lifecycle (planning). A search for information of interest was performed (action). The analysis of the information (observation) enabled us to discover that the object of study presented a high level of complexity since several aspects of different natures had to be taken into account. The reasoning about these elements (reflection) allowed us to realize that the possible solutions should be made up of the integration of several solutions to partial problems, that is, defining the MINERVA framework general structure as a collection of three different types of elements: conceptual, methodological and tool support. This cycle then included a comprehensive study of the different aspects that influence the implementation of BPs by means of services, model-driven approaches, modeling and metamodeling of BPs and services, continuous process improvement, conceptual architectures for this kind of frameworks and for the integration of software tools, etc.
- General cycles: For each of the three types of elements defined in MINERVA framework (conceptual, methodological, tool support) general cycles were defined to be able to answer the following questions and to define the different components of the framework based on the answers:
  - Conceptual: what is needed to manage the complexity of BPM in organizations? How can the needed information be represented in the framework?
  - Methodological: which methods/techniques are useful for managing the work throughout the framework? How may services be derived from BP models? How can the continuous process improvement based on the BP lifecycle defined with focus on execution measurement be guided?
  - Technological: which software tools are useful for supporting the work throughout the framework? Which existing tools can be integrated into the framework and which tools do we need to develop on purpose?

### 2.2.2. Experiments

An experiment, described in chapter 10 was carried out to validate the Suitability of the QVT transformations (i.e. the generation of SoaML service models from BPMN2 models) and the Understandability of the results of the QVT transformations (i.e. the generated SoaML service models) as defined by ISO 9126 [ISO, 2001].

### 2.2.3. Case study

Two case studies, which are described in chapter 10 were carried out to validate the proposals in MINERVA framework, as the IT area of the HGCR could not participate in the project in the end, so the methodology and the model-driven approach to generate services from BP models were validated in the context of another organization.

#### **2.2.4. Systematic review**

At the beginning of this thesis work a systematic review of the literature was carried out as described in chapter 3 in order to find existing proposals related to the application of SOC and MDD paradigms to BPM.

### **2.3. Conclusions**

In this Chapter research methods in Software Engineering were presented along with their use within this thesis work: Action-Research to guide the definition of the MINERVA framework and its elements (described in this chapter), experiments and case studies (described in chapter 10) as empirical methods to validate the proposals in the framework, and a systematic review process which was applied at the beginning of this thesis in an effort to find related work on the subject (described in chapter 3).

The use of the research methods presented here allowed us to follow a systematic method for the realization and validation of the work in this thesis, providing the guidelines and support needed to make the results more reliable and ensuring that there was the basis set for the replication of the validations performed on the proposals.



## Chapter 3.

### State of the art

This Chapter describes the state of the art in several topics that this thesis deals with, presenting concepts, definitions and related work.

The Chapter is organized as follows: Business Process Management (BPM) is described in section 3.1, Service Oriented Computing (SOC) in section 3.2, Model Driven Development (MDD) in section 3.3 and Continuous Process Improvement (CPI) in section 3.4, including key standards associated with them. In section 3.5 the systematic literature review carried out when starting this thesis is presented for the integration of the BPM, SOC and MDD paradigms, and finally in section 3.6 conclusions for the chapter are discussed.

The contents of this chapter are used throughout the rest of the chapters of this thesis, as they provide the basis for all definitions and elements in MINERVA.

#### 3.1. Business Process Management (BPM)

This section presents the main elements involved in the Business Process Management (BPM) paradigm to manage BPs throughout their lifecycle. In the first place, definitions and concepts for BPM and BPM Systems to support it are provided, secondly BP concepts and lifecycle are described including process patterns, and finally, three main standards for BPs are presented: BPMN2 OMG [2011a], XPDL WfMC [2008] and WS-BPEL OASIS [2007].

##### 3.1.1. BPM and BPMS

Business Process Management (BPM) refers to the set of activities that organizations carry out to optimize or adapt their business processes to new organizational needs BPMI [2000-2005] whose main focus is on business processes (BPs) in organizations. In [van der Aalst et al., 2003b] BPM is defined as “supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information”. For [Weske, 2007] BPM “includes concepts, methods, and techniques to support the design, administration, configuration, enactment and analysis of BPs”, where the basis for BPM is the explicit representation of BPs, which is the one adopted in this thesis.

It could be said that a key breakthrough point was at the beginning of the nineties by means of the publication of two articles: one was by Davenport [Davenport and Short, 1990] and another was written by Hammer [Hammer, 1990], reporting on process innovation and radical process change. BP concepts can, however, be traced back to the early twenties under the term “methods and procedures analysis” [Smith and Fingar, 2003]. As is stated in [Smith and Fingar, 2003] “Companies have always searched for new ways of restructuring work and improving business organizations, but until very recently a practical way to implement and manage the lifecycle of BP design and execution was seriously lacking”.

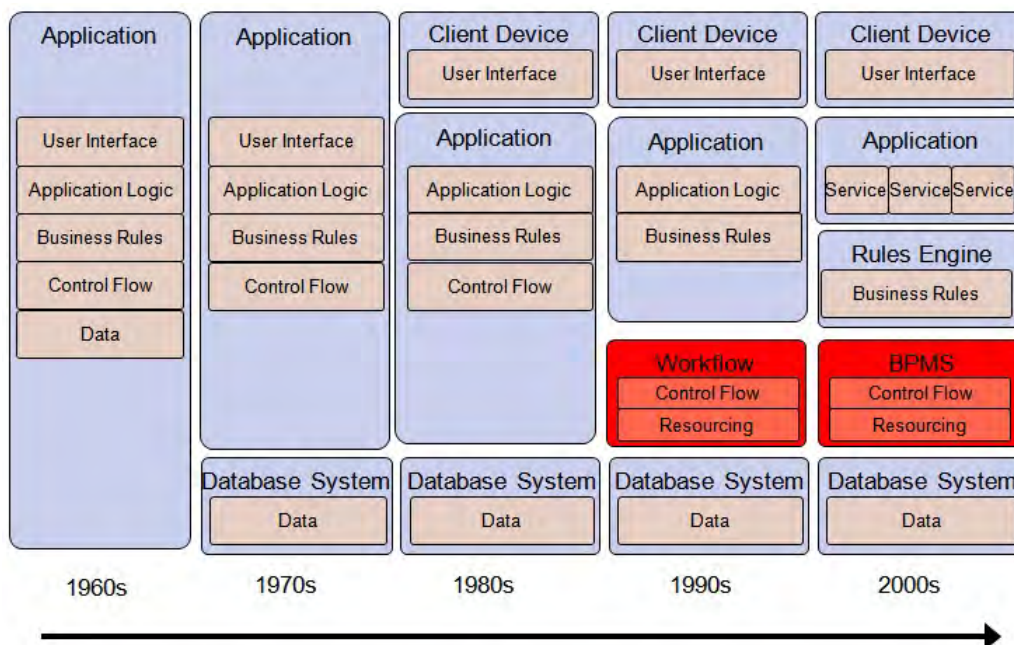
The different views, terminology and ways of working existing between the business and IT areas, from the business manager and business analyst to the systems analyst and programmer, had been

an obstacle to the successful support of BPs by IT systems and for the introduction of changes when required by the business [Smith and Fingar, 2003]. Several attempts to provide support for an integrated view of organizations included Enterprise Resource Planning (ERP) systems, which provide an integrated database for the entire organization, Enterprise Architecture Integration (EAI) efforts, Supply Chain Management (SCM) and Customer Relationship Management (CRM) systems.

BPMS are a new kind of IT systems that allow organizations to manage, measure, change and control their BPs by means of a single and centralized system providing all functionalities needed around the explicit representation and execution of BPs. For [van der Aalst et al., 2003b] a BPMS is “a generic software system that is driven by explicit process designs to enact and manage operational business processes”, which is also defined in [Weske, 2007] as “a generic software system that is driven by explicit process representations to coordinate the enactment of BPs”, which is the one adopted in this thesis.

BPMS provide the means for managing BPs throughout the BPs lifecycle, with a single definition of BPs in BP models from which different views can be derived and IT systems can be built, shifting from “data processing” to “process processing” [Smith and Fingar, 2003]. BPMS provide the needed IT support for organizations and business area by “combining aspects of workflow, process automation and transaction management to provide a global visibility and control .. ” in process-aware systems that are understood, used and changed by both business and IT areas working together [Smith and Fingar, 2003]. In [van der Aalst et al., 2003b] the same is stated “systems supporting BPM need to be “process aware”, i.e., without information about the operational processes at hand, little support is possible”.

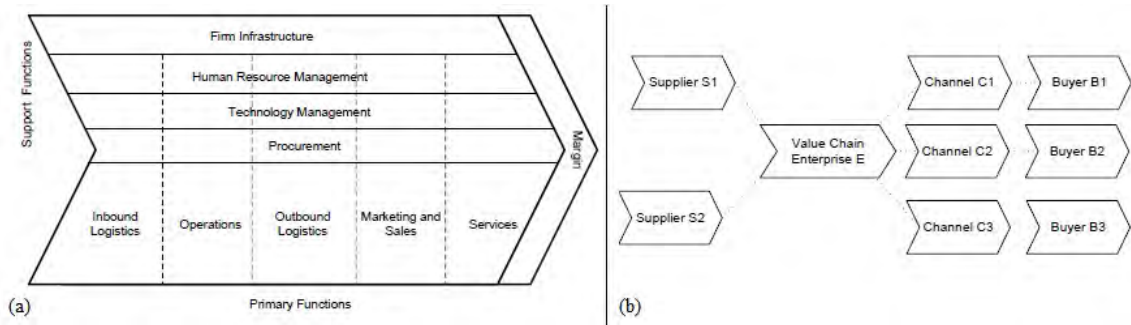
A Process-Aware Information System (PAIS) is defined in Dumas et al. [2005] as “a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models”. This definition clearly includes BPMS and also other tools providing support for BPs such as SAP and workflow management systems. From the sixties where applications were monolithic to the present state of technology, several advances had been made to provide support for BPM, as shown in Figure 3.1.



**Figure 3.1.:** Evolution of BPM technology from [ter Hofstede et al., 2009]

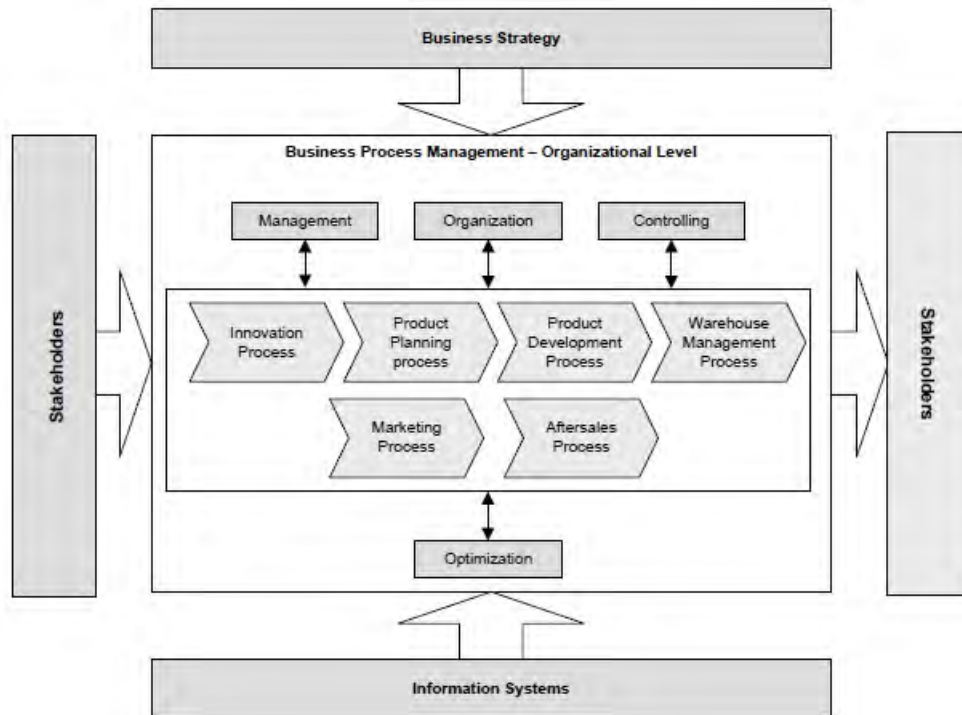
In addition to technological developments “business administration also contributed to the rise of BPM” with two factors, mainly: value chains for functionally breaking down activities in an organization and process orientation to organize them [Weske, 2007]. Value chains, developed by

Michael Porter, are a way to organize business functions relating them to each other, providing a high-level view of how an organization operates, where all functions have to contribute to the success of the business. These functions can be partitioned into primary and support functions, where the first contribute directly to the competitive advantage of the organization and the second provide the support for the first [Weske, 2007]. A value system shows how several value chains of different organizations are related in order for them to cooperate with each other. In Figure 3.2 the internal structure of a value chain (a) is shown, along with a value system (b) from the point of view of the organization (E).



**Figure 3.2.:** (a) Value chain and (b) value system for organization (E) from [Weske, 2007]

“At an organizational level, process orientation has led to the characterization of the operations of an enterprise using business processes” [Weske, 2007] where the structure of BPM at the organizational level comprises four main elements: the BPM space, the business strategy, information systems and stakeholders. The business strategy defines the goals of the organization, the information systems are assets of the organization which provide workers with the support needed to perform their work; stakeholders include external business partners, customers and employees in the organization; they all influence and are influenced by, the BPs in the organization [Weske, 2007]. In Figure 3.3 the organizational-level BPM for an organization is shown.



**Figure 3.3.:** Organizational-level BPM from [Weske, 2007] based on Schmelzer and Seselmann (2006)

Key elements for carrying out BPM in organizations are the BPs along with the BP lifecycle which guides the different activities to be supported through BPM efforts, which are discussed in the next section.

### 3.1.2. BP Concepts and lifecycle

In this section what a BP means is discussed in the first place, describing several related concepts such as BP model and BP instances, as well as different types of BPs, and process patterns for BP modeling. After these, the BP lifecycle is presented describing each of its phases.

#### 3.1.2.1. BP definitions

For an organization to achieve its defined business goals in an efficient and effective manner, people and other organizational resources, such as information systems must play together well, where BPs are a central element to facilitate this collaboration. BPs are essential to understand how an organization works at an organizational level, but they are also important to provide the basis for the design and implementation of systems to support them [Weske, 2007]. BPs were defined at the beginning of the nineties by [Hammer, 1990, Hammer and Champy, 1993] as “a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer”, which has no explicit implications about the ordering of activities and puts emphasis on the input/output view of BPs, with a focus on delivering value to the customer.

Another definition given at the same time by [Davenport and Short, 1990, Davenport, 1992] is “a set of logically related tasks performed to achieve a defined business outcome for a particular customer or market... a specific ordering of work activities across time and place, with a beginning, an end and clearly identified inputs and outputs” in which the ordering of activities is specified while defining inputs/outputs and focusing on the customer or market. The kinds of interactions a BP can present are also specified, as BPs “have customers (internal or external) and they cross organizational boundaries, i.e. they occur across or between organizational subunits”, providing a complete view on the nature of BPs.

More recently, and based on those definitions [Weske, 2007] states that “a BP consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each BP is enacted by a single organization, but it may interact with BPs performed by other organizations”, which is the one adopted in this thesis.

The main characteristics of BPs include that they are large and complex, they involve several sections or areas from the organization and from different organizations, and their execution can last for weeks, months and even years. Apart from these features, there are specific BPs for several domains such as health or finances, they can have partial or total systems support, and they are very difficult to make explicit as they are mainly embedded in existing software systems.

BPs are specified as BP models which “consists of a set of activity models and execution constraints between them” and these models when executed have different BP instances which “represents a concrete case in the operational business of a company, consisting of activity instances” [Weske, 2007]. A BP model corresponds to the specification of a BP in the organization that has a unique identifier and “represents a blueprint for a set of process instances with similar structure” [Weske, 2007].

On the other hand, BP cases correspond to a specific occurrence of the execution of the model, including the particular data associated with the activities executed for that instance. One BP model is associated with many BP cases and each BP case is associated with only one BP model. In the same way, an activity model represents the definition of an activity that has a unique identifier and is associated with a BP model, and an activity instance corresponds to each occurrence of an activity in a BP case. In Figure 3.4 a simple BP model for a Reseller is shown, along with two example BP instances.



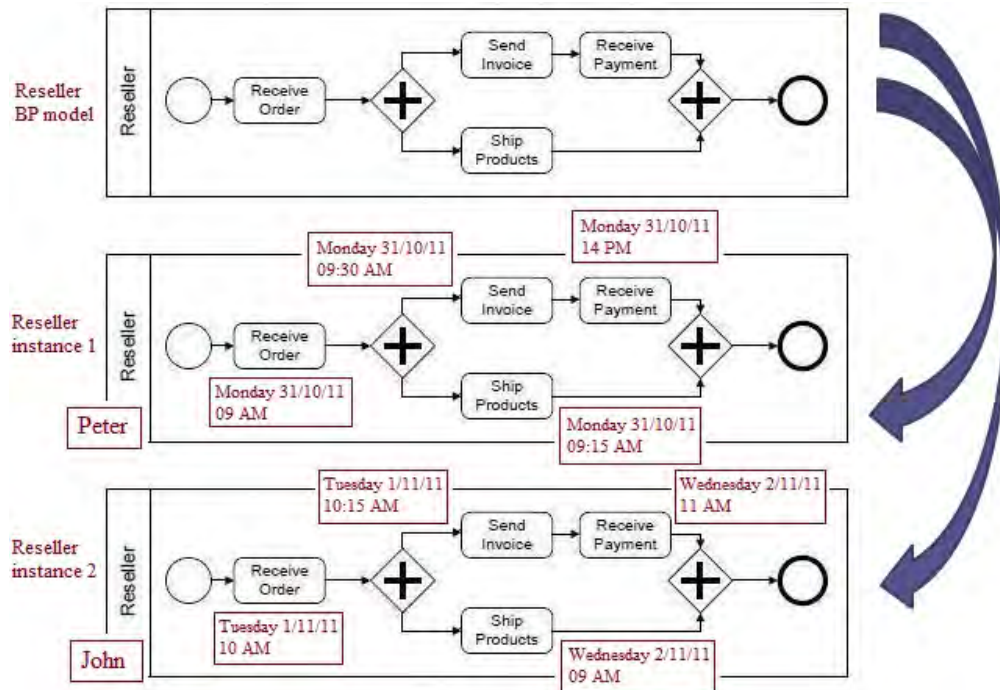


Figure 3.4.: BP model and BP instances based on [Weske, 2007]

A common classification of BPs is performed based on the location of the control of the execution flow in orchestration and choreography [Weske, 2007], the latter also being referred to as collaborative BPs which include the choreography [OMG, 2011a], which is the one adopted in this thesis. In an orchestration, the control is owned and is centralized by a single organization performing the BP, in analogy with an orchestra director who controls its musicians. In a collaborative BP, there is no central control for the realization of the BP; it is arranged and spread between the participants in the BPs, interacting by means of messages, in analogy to dancers who agree on a common choreography to be executed in the show [Weske, 2007]. In Figure 3.5 an example of an orchestration and a collaborative BP is shown.

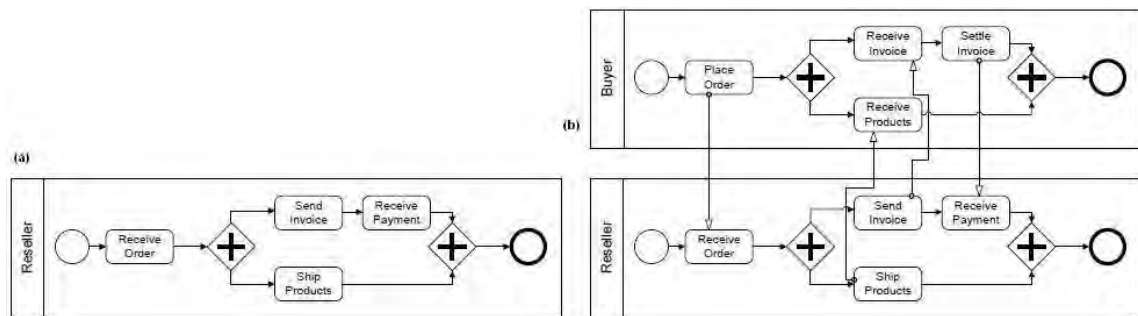


Figure 3.5.: (a) orchestration and (b) collaborative BP from [Weske, 2007]

In Figure 3.5 it can be observed that in the orchestration (a) the process is defined within the limits of one organization, in the example this is the Reseller, while in the collaborative BP (b) there are two different processes defined within the limits of each participant organization, the Buyer and the Reseller, who interact by means of messages.

### 3.1.2.2. BP patterns

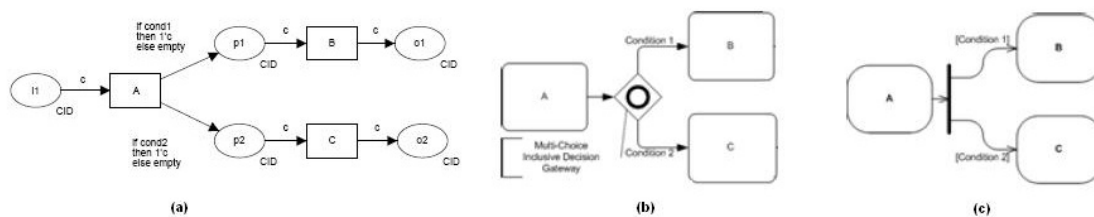
A pattern can be informally defined as a general solution (reusable) for a recurrent problem in a given context, describing the problem, the solution, when to apply it and the consequences

of doing that. The idea of patterns was first conceived by the Architect Christopher Alexander (Vienna, Austria) for architectural designs (1977), and it was adapted to many disciplines including Software Engineering in the form of design patterns at different abstraction levels: architectural design (POSA, SOA), OO design (GoF Gamma et al. [1995]), and idioms for languages.

BP patterns (workflow patterns<sup>1</sup> as they are called) propose several modeling solutions to general modeling problems that occur when specifying BP models. In contrast to the GoF patterns which are documented as object models and code recipes, the BP patterns are mostly visual, showing the right arrangement of BP elements (activities, gateways, flows) to solve the problem detected [Havey, 2005]. BP patterns are independent of any modeling language and can be expressed in several existing ones, serving also as a way to compare their expressiveness [Weske, 2007].

BP patterns are defined taking into account the perspectives as defined in [Jablonski and Bussler, 1996] including: control flow van der Aalst et al. [2003a], Russell et al. [2006a], data [Russell et al., 2005a] and resources [Russell et al., 2005b], as well as for exception handling [Russell et al., 2006b]. BP patterns are an essential part of BP modeling as they define and standardize how to solve modeling problems that appear when a specific behavior is to be represented. The control flow perspective is essential for the specification of the BP model, as the activities to be executed and their precedence order, as well as possible bifurcations of execution, are defined in this perspective.

Control flow patterns cover from very simple constructions such as the sequence pattern to very complex routing ones that are not supported in most BPMS. For the simpler patterns a description of the pattern, a graphical representation (in Petri Nets), synonyms and examples are provided; for the more complex patterns a description of the problem (indicating why it is difficult to realize in current systems) and potential solutions (how to model it combining basic routing constructions) are provided. In Figure 3.6 the control flow pattern Multiple choice is presented as an example in: (a) Petri Nets from<sup>1</sup>, (b) BPMN and (c) UML AD, both from [White, 2004].



**Figure 3.6.:** Multiple choice control flow pattern (a) Petri Nets (b) BPMN and (c) UML AD

In Figure 3.6 the Multiple Choice control flow pattern is presented, whose meaning is as follows: once activity *A* is executed, the execution of activities *B* and *C* is determined by the satisfaction of the conditions specified for each branch, the possible execution being only *B*, only *C* or both, as this pattern corresponds to the logic operator OR. BP patterns are also a way to ensure the correctness and quality of BP models regarding the modeling of selected parts as stated in the defined patterns.

Several comparisons on BP patterns support for different BP modeling languages and tools can be seen in the workflow patterns initiative, and a comparison for the first twenty control flow patterns between BPMN and UML AD is presented in [White, 2004]. In MINERVA BP patterns are one of the recommended guides for modeling BPs.

### 3.1.2.3. BP lifecycle

BPM and BPMS in organizations support all activities defined in a reference BP lifecycle, which guides the management of BPs from modeling and implementation to execution and evaluation. There are several and similar proposals for BP lifecycles such as [van der Aalst, 2011] proposing five phases of: (re)Design, Configuration/implementation, Enactment/monitoring, Adjustment, and

<sup>1</sup><http://www.workflowpatterns.com/>

Diagnosis/requirements, and the one adopted in this thesis by [Weske, 2007], proposing four phases of: Design&Analysis, Configuration, Enactment and Evaluation, which is shown in Figure 3.7.

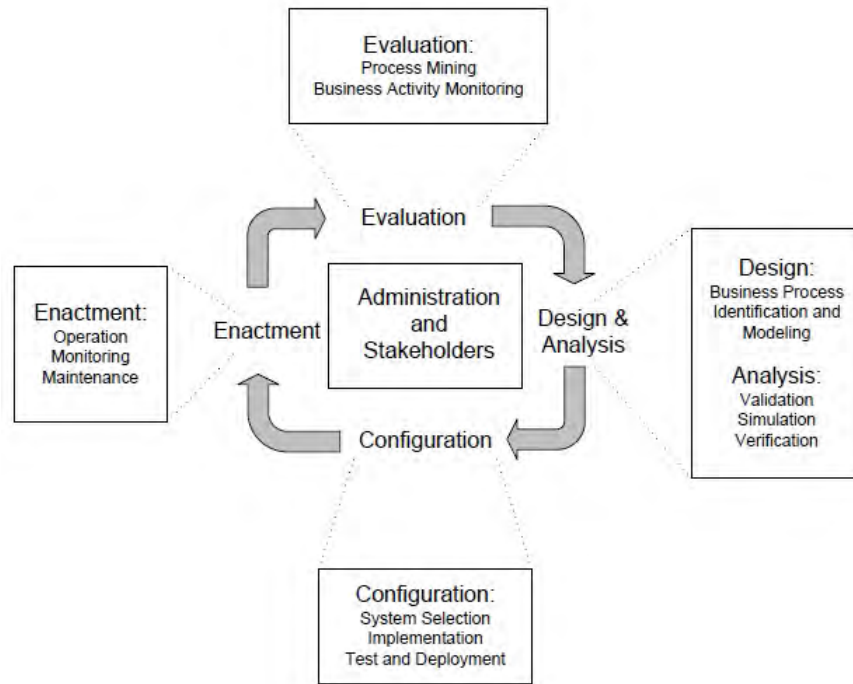


Figure 3.7.: BP lifecycle from [Weske, 2007]

The cycle structure shows the logical dependencies between the defined phases, but several activities can be performed with an incremental and evolutionary approach.

The BP lifecycle is entered in the **Design&Analysis phase**, in which the main focus is on identifying and explicitly modeling BPs and validating them. As stated in [Weske, 2007] “explicit BP models expressed in a graphical notation facilitate the communication about these processes, so that different stakeholders can communicate efficiently, and refine and improve them”. With the participation of the business area, BPs are identified, reviewed, validated and specified as BP models in the selected notation, such as Business Process Model and Notation (BPMN2), Unified Modeling Language (UML), Event-driven Process Chain (EPC), Petri Nets or Yet Another Workflow Language (YAWL). The BP models generated are validated with business people by means of analytical techniques or simulation tools that allow to execute several and different configurations for the BP in order to get insight into their execution; and are also verified in order to detect undesired characteristics such as deadlocks.

The **Configuration phase** refers to the selection of configurations for the implementation of the modeled BPs. Once the model is in place, the process can be implemented in different ways: with procedures and policies that can be followed in the organization without software support, with one or more systems supporting the BPs, or with integrated systems for BPM (BPMS). Then the implementation is tested by means of traditional testing techniques, such as functional testing of the behavior (i.e. activities and control flow) and other tests such as integration and performance, attempting to detect possible execution problems.

In the **Execution phase** BPs are deployed in the selected infrastructure, and BP instances occurred as they are initiated. BP instances execution is monitored to provide information on their status (generally with visualization techniques based on colors i.e. green for enabled activities, blue for running instances). Data on BPs execution is collected and stored typically in some form of log file in which events that have occurred are registered. These events typically refer to start and end of activities, among others, and constitute the basis for the next phase in the lifecycle.

In the **Evaluation phase** the objective is to evaluate and improve BP models and their implementation based on the information registered in the log files, by means of techniques such as Process

Mining or Business Activity Monitoring (BAM). Process Mining is an active field research, which presents different applications such as discovering BP models from BPs execution when they do not exist, conformance checking between existing BP models and real BP execution, and extending BP models with information from BPs execution. Business activity monitoring can be used to detect, for example, shortage of resources for executing activities, and as this is also useful for BP simulation, these phases are strongly related.

Administration procedures are defined to manage BP models and information on BP instances among other artifacts from the BP lifecycle, such as a BP models repository with query mechanisms. Several types of stakeholders must be classified and represented based on their knowledge, expertise and experience, such as process designer or process participant.

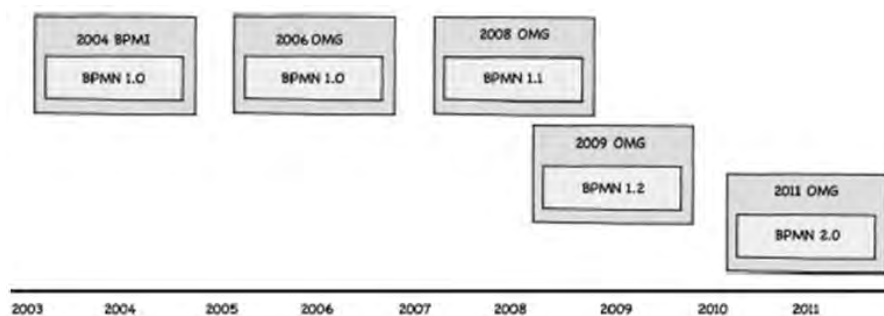
### 3.1.3. Main standards for BPs

In this section three main standards related to BP modeling and execution are presented: BPMN2 [OMG, 2011a] for BP modeling which includes the notation previously presented and other elements presented here, XML Process Definition Language (XPDL) [WfMC, 2008] which was specifically designed for workflows execution and Web Services Business Process Execution Language (WS-BPEL) [OASIS, 2007] specifically designed for BP execution based on WS interactions. These three standards are integrated in MINERVA and used throughout this thesis.

#### 3.1.3.1. Business Process Model and Notation (BPMN2)

The first version of BPMN came from the business area [BPMI, 2000-2005] and was released as an (OMG) <sup>2</sup> standard in 2006, after BPMI and OMG merged, by the name of Business Process Modeling Notation (BPMN). At that time the only elements included in BPMN were the ones for specifying the notation, and soon a separated standard was defined providing a metamodel for specifying BP notations, including BPMN, the Business Process Definition Metamodel (BPDM) [OMG, 2008a]. The semantics for each of the defined elements in the notation were not completely defined, so this BPMN was not executable, although at the end of the standard a partial mapping to WS-BPEL was provided for execution.

Several research and industrial initiatives were carried out to provide complete transformations from BPMN to WS-BPEL, also in addition successive versions of the XPDL standard were aligned to the definitions in BPMN so transformations from BPMN to XPDL were straightforward, making XPDL the preferred exchange format for BPMN models. The BPMN2 version of the standard adds several important changes to the previous versions: first of all, the corresponding metamodel was included in the standard and the execution semantics for the elements defined by the notation were specified, making it executable; secondly, both XML Metadata Interchange (XMI) and XML formats were also defined for the exchange of BP models including diagrams, providing the means for different tools implementing the standard to interoperate with each other. In Figure 3.8 the evolution of the BPMN standard is presented.



**Figure 3.8.:** BPMN standard evolution from [Rademakers and van Liempd, 2011-2012]

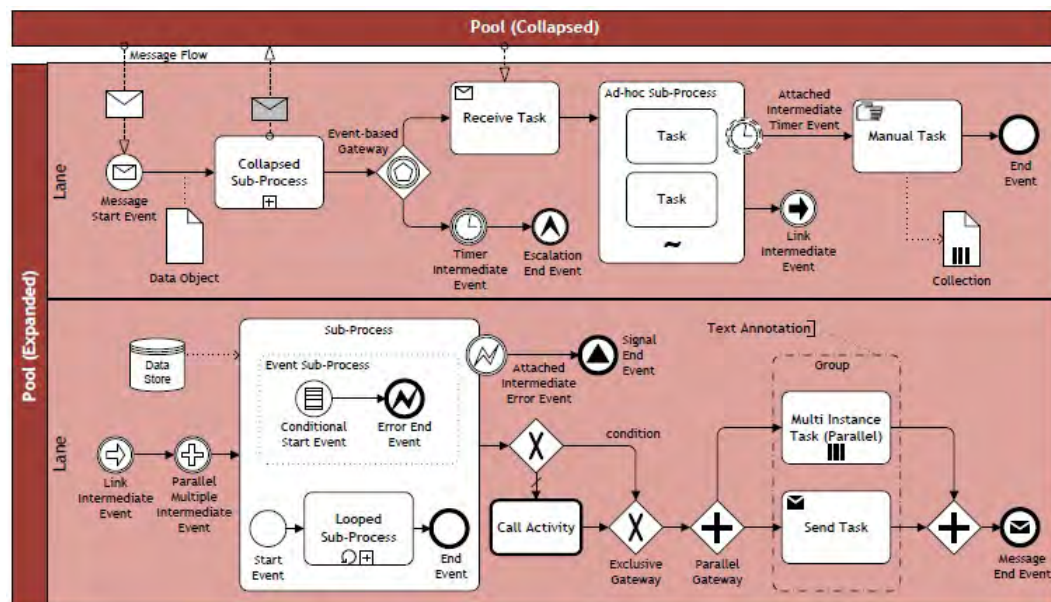
<sup>2</sup><http://www.omg.org>

As for the notation for modeling BPs, BPMN2 provides several elements that were formerly grouped in four main categories: flow objects, connection objects, swimlanes and artifacts (in BPMN2 data is separated) but in BPMN2 these categories are not completely preserved as the structure of the standard was changed to a layered style. It provides a core set of elements with which it is possible to model most BPs, upon which the rest are constructed.

Activities (tasks of different types, sub-processes, both with markers such as multi-instance), gateways (AND, XOR, OR, Complex) and events (start, intermediate and finish, of different types such as messages or compensation) are defined as Flow objects. A task defines an atomic unit of work representing the job to be performed and can not be further broken down; when this is possible, a sub-process is defined containing several other elements. Events are something that occur during the execution of the process, at its start, in the middle of the execution or when it finishes. Gateways represent the divergence and convergence of the flow between elements in the process, defining the possible paths to be taken in the execution of the process.

Connection objects included sequence flows (to be used within a pool) and message flows (to be used within different pools) but in BPMN2 this category is not shown. Swimlanes defines pools to denote participants in a collaborative process and lanes as sub-partition of pools for sections, areas or roles, among others. Artifacts provide textual annotations, a mark for grouping sub-parts of the process only for decoration, and associations (to be used between artifacts or data, and flow objects); and data provides different types of data and message objects which also serve to decorate the model.

In Figure 3.10 some of the main elements defined in BPMN2 are shown adapted from the BPMN2 poster<sup>3</sup> of the BPMBerlin offensive. It also defines three different types of processes: orchestration, choreography and collaboration (including a conversation view) to model BPs within an organization and between two or more organizations, providing two different views of message interactions (collaboration to show the pools and activities involved within each pool, and choreography to focus on messages between the pools). Conversations provide a high level view of the interactions between different participants. In Figure 3.9 an example of a collaborative BP in BPMN2 is shown from the BPMN2 poster of BPMBerlin.



**Figure 3.9.:** BPMN2 collaborative BP example from the BPMN2 poster of BPMBerlin

In Figure 3.9 the use of several elements from the notation can be seen, in a collaborative process between two participants which are modeled in two pools; the upper pool is collapsed meaning that we are not aware of the process from this participant; the lower pool is expanded showing the flow and elements in the process.

<sup>3</sup><http://www.bpmb.de/index.php/BPMNPoster>

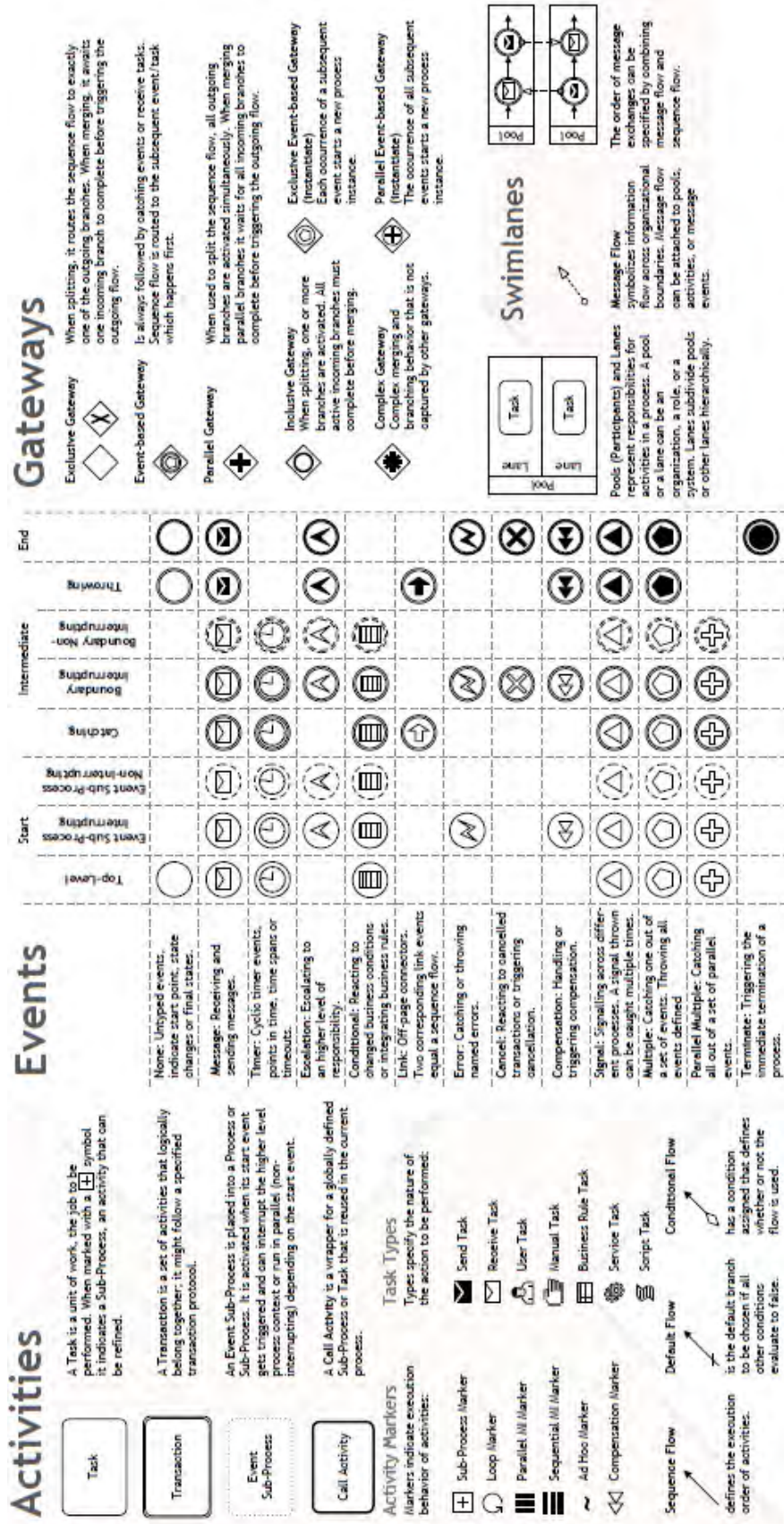


Figure 3.10.: BPMN2 elements (some) from the BPMN2 poster of BPMBerlin

As for the standard view, BPMN2 is structured in layers where each layer builds on top of and extends lower layers [OMG, 2011a], the innermost layer being the BPMN core which contains three sub-packages: foundations defining fundamental constructions for BP modeling, Commons including elements that are used from Process, Choreographies and Collaborations, and Services providing constructions for service and interfaces modeling. Upon these the Process, Choreographies and Collaborations are built defining the elements contained in them. In addition for Process, Activities, Data and Human are included, and for Collaborations, Conversations are added. In Figure 3.11 the BPMN2 structure is shown, along with the elements defined in the BPMN Core.

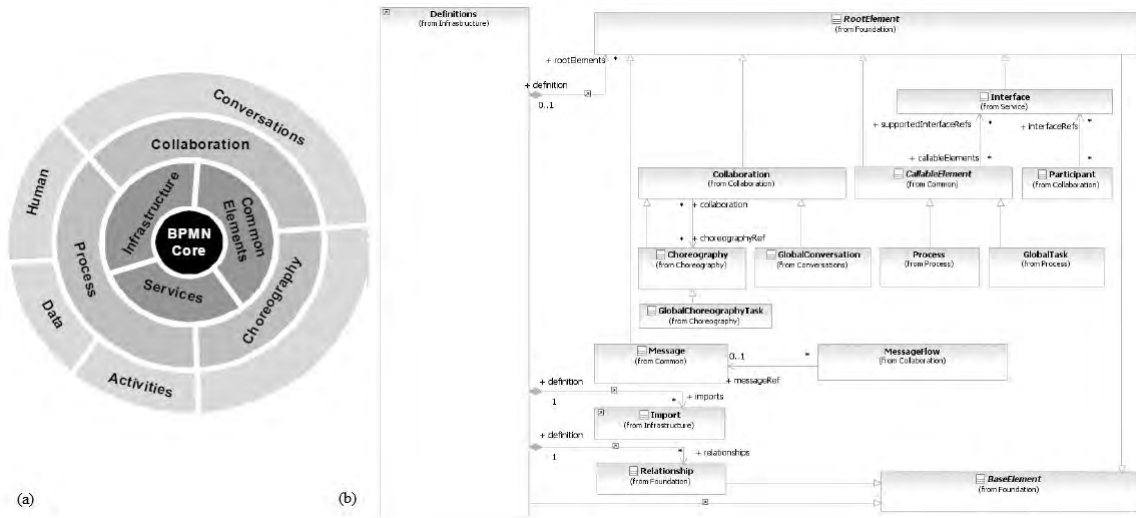


Figure 3.11.: BPMN2 layers structure and BPMN Core elements from [OMG, 2011a]

The definitions element is the outermost containing object for all BPMN2 objects, where the RootElement is the abstract super class for all BPMN elements that are contained within Definitions. Examples of concrete RootElements include Collaboration, Process, and Choreography [OMG, 2011a]. In Figure 3.12 an example of the structure of a BPMN2 XML file is presented, showing the definition of a Collaboration and one of the Process involved and its elements.

```

[?xml version="1.0" encoding="UTF-8" standalone="yes" ?]
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="oryx_2b979278-22f1-40d7-bcab-e870dd634fb9" name="Vendedor"
targetNamespace="http://www.omg.org/bpmn20">
  <process isClosed="false" name="Entidad de credito" id="oryx_ce326c75-a417-453b-9264-938d644741b5">
    <laneSet name="Entidad de credito" id="oryx_85E20A47-42BD-46BD-B6D0-5A99B9AECCE2">
      <lane id="oryx_088DA330-0E52-4D69-A374-0CF751265FD6" name="Entidad de credito">
        <flowNodeRef>oryx_283CF4A9-33A4-4567-BD36-941A9D7AEC4</flowNodeRef>
        <flowNodeRef>oryx_5424B6BD-DC4B-4DFF-84EE-50B4412727BA</flowNodeRef>
        <flowNodeRef>oryx_A7E00E0B-F9B5-404C-A0D1-7A34BDAFC502</flowNodeRef>
        <flowNodeRef>oryx_C5B04590-2F64-4124-B7FC-A684CAF43F79</flowNodeRef>
      </lane>
    </laneSet>
    <startEvent name="" id="oryx_283CF4A9-33A4-4567-BD36-941A9D7AEC4"/>
    <endEvent name="" id="oryx_5424B6BD-DC4B-4DFF-84EE-50B4412727BA"/>
    <serviceTask completionQuantity="1" startQuantity="1" isForCompensation="false" name="Recibir o:
    <task completionQuantity="1" startQuantity="1" isForCompensation="false" name="Enviar confirmac:
    <sequenceFlow targetRef="oryx_A7E00E0B-F9B5-404C-A0D1-7A34BDAFC502" sourceRef="oryx_283CF4A9-33
    <sequenceFlow targetRef="oryx_C5B04590-2F64-4124-B7FC-A684CAF43F79" sourceRef="oryx_A7E00E0B-F9
    <sequenceFlow targetRef="oryx_5424B6BD-DC4B-4DFF-84EE-50B4412727BA" sourceRef="oryx_C5B04590-2F
  </process>
  <collaboration id="oryx_93d5ec57-e778-44df-866c-cd5892ac9270" name="Vendedor">
    <participant processRef="oryx_a72b10cc-9d60-404c-bccc1-a388e214973c" name="Comprador" id="oryx_5:
    <participant processRef="oryx_id7f3061-b45c-41cc-b15e-ee62f48977e1" name="Vendedor" id="oryx_04:
    <participant processRef="oryx_ce326c75-a417-453b-9264-938d644741b5" name="Entidad de credito" id=
    <messageFlow targetRef="oryx_6c848c94-A123-4017-AD12-B434C998DCEB" sourceRef="oryx_354c9366-260:
    <messageFlow targetRef="oryx_46775851-EAF0-4F68-8F0E-401BE8E38E02" sourceRef="oryx_25C346F9-8B6:
    <messageFlow targetRef="oryx_DA7856E9-FA2F-4612-9293-2BA88C183562" sourceRef="oryx_F248424B-5CC:
    <messageFlow targetRef="oryx_A7E00E0B-F9B5-404C-A0D1-7A34BDAFC502" sourceRef="oryx_C148281B-380:
    <messageFlow targetRef="oryx_45C26702-3996-4E16-B4C6-61CE37E00193" sourceRef="oryx_C5B04590-2F6
  </collaboration>
</definitions>

```

Figure 3.12.: BPMN2 XML file example

In Figure 3.12 it can be seen that sequenceFlow and messageFlow elements are both defined by means of a targetRef and a sourceRef which references the elements they connect, in the first case between the elements defined in the same process and in the second between elements defined in different processes. In the case of the Process it also can be seen that the four elements defined in the Process (startEvent, endEvent, servicetask and task) are contained in the defined lane as flowNodeRef. In the case of the Collaboration the participants involved and the messageFlows between them are the only elements needed to define it in this case, as the participants contain the reference processRef to each process defined. In Figure 3.13 the metamodel corresponding to the Collaboration element is presented, as a key element in this thesis work, showing other elements such as Conversation and Choreographies that can be part of, or referenced from, a Collaboration.

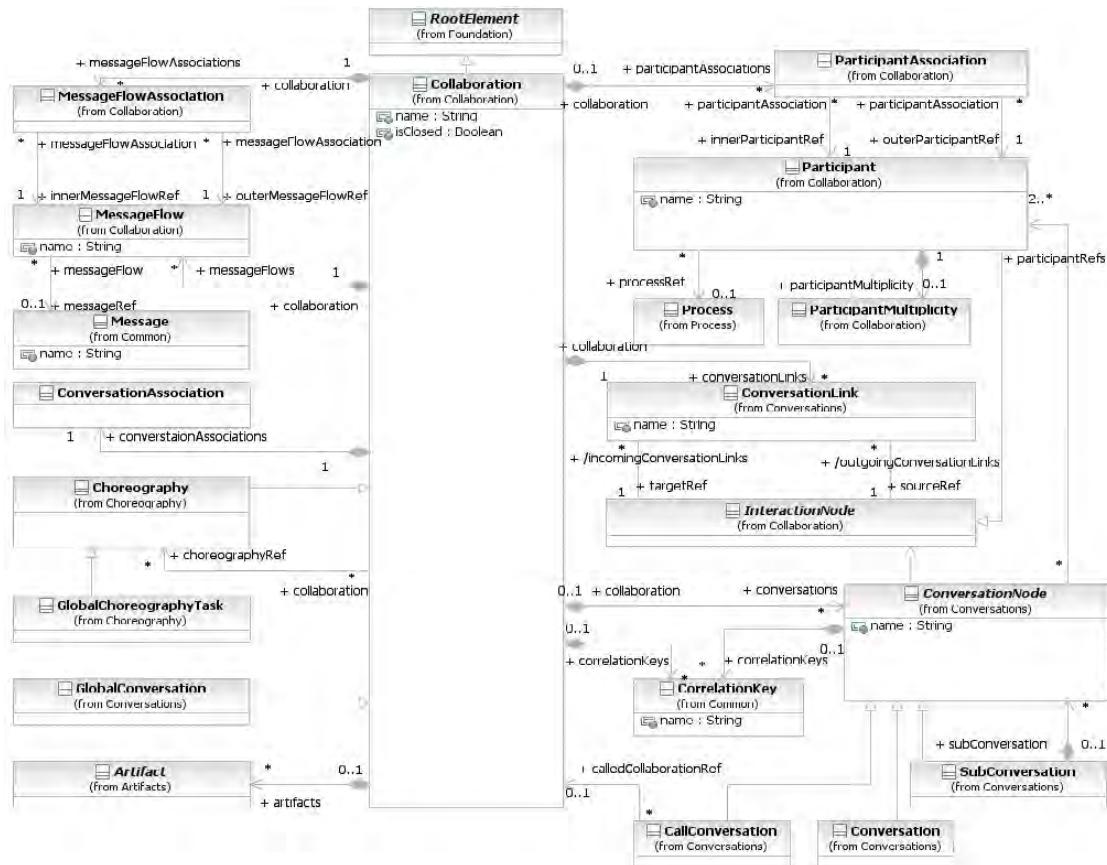


Figure 3.13.: BPMN2 metamodel for Collaborations from [OMG, 2011a]

The Services sub-package provides elements to model services such as the definition of interfaces with operations and messages in and out from them, while connecting them to elements inside the defined processes by means of the reference InterfaceRefs included in the participants element, which in turn corresponds to a Process. For the execution of services several other elements have to be included in the BPMN2 model such as input and output data and references to the implementation (i.e. WS). In Figure 3.14 an example of WS invocation from a ServiceTask is shown providing all the elements needed and the reference to the WSDL defining the WS.



```

<import importType="http://schemas.xmlsoap.org/wsdl/"
location="http://localhost:8181/pruWS/Echo2Service?wsdl"
namespace="http://echo/" />
<message id="echo" itemRef="tns:echoItemRequest" />
<message id="echoResponse" itemRef="tns:echoItemResponse" />
<itemDefinition id="echoItemRequest" structureRef="echo:echo" />
<itemDefinition id="echoItemResponse" structureRef="echo:echoResponse" />
<!-- Interface: implementationRef = QName of WSDL Port Type -->
<interface name="Echo Interface" implementationRef = "echo:Echo2" >
  <!-- Operation: implementationRef = QName of WSDL Operation -->
  <operation id="echoOperation" name="Echo Operation" implementationRef="echo:echo">
    <inMessageRef>tns:echo</inMessageRef>
    <outMessageRef>tns:echoResponse</outMessageRef>
  </operation>
</interface>
<itemDefinition id="input" structureRef="string" />
<itemDefinition id="arg0" structureRef="string" />
<itemDefinition id="return" structureRef="string" />
<itemDefinition id="output" structureRef="string" />
<serviceTask id="webServiceTask" name="Call WS" implementation="##WebService"
operationRef="tns:echoOperation">
  <!-- The BPMN 2.0 Meta Model requires an Input/Output Specification -->
  <iOSpecification>
    <dataInput itemSubjectRef="tns:echoItemRequest" id="dataInputOfServiceTask" />
    <dataOutput itemSubjectRef="tns:echoItemResponse" id="dataOutputOfServiceTask" />
    <inputSet>
      <dataInputRefs>dataInputOfServiceTask</dataInputRefs>
    </inputSet>
    <outputSet>
      <dataOutputRefs>dataOutputOfServiceTask</dataOutputRefs>
    </outputSet>
  </iOSpecification>
  <dataInputAssociation>
    <sourceRef>input</sourceRef>
    <targetRef>arg0</targetRef>
  </dataInputAssociation>
  <dataOutputAssociation>
    <sourceRef>return</sourceRef>
    <targetRef>output</targetRef>
  </dataOutputAssociation>
</serviceTask>

```

Figure 3.14.: BPMN2 XML file for WS invocation from ServiceTasks

The data for the graphical diagram is added at the bottom of the BPMN2 file to allow different tools to be able to load the graphical BP model as it was originally defined, by means of the BPMN DI (Diagram Interchange) and DC (Diagram Commons) defining elements to be represented. The basic concept is that serializing a diagram [BPMNDiagram] for exchange requires the specification of a collection of shapes [BPMNShape] and edges [BPMNEdge] on a plane [BPMNPlane] [OMG, 2011a]. In Figure 3.15 an example of a BPMN diagram in a BPMN2 file is shown.

```

<bpmndi:BPMNDiagram id="sid-ab519311-4fa2-4daf-953d-d181144e3b18">
  <bpmndi:BPMNPlane bpmnElement="sid-0ffe4fe6-d227-4d35-853b-6fcf37d9c788" id="sid-04e">
    <bpmndi:BPMNShape bpmnElement="sid-37ADF273-8DF2-4A34-BEB4-6221EBE93652" id="sid-
    <omgdc:Bounds height="214.0" width="722.0" x="0.0" y="225.0"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="sid-70B5ADBA-15DF-4EE1-A002-CBD9C73CB8E9" id="sid-
    <omgdc:Bounds height="213.0" width="723.0" x="0.0" y="0.0"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="sid-CCD36BAF-E48F-439C-82B5-58D53560D6E0" id="sid-
    <omgdc:Bounds height="150.0" width="721.0" x="0.0" y="450.0"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="sid-0DD2CE05-FF0D-48A8-BF4C-A6FE280D4E77" id="sid-
    <omgdc:Bounds height="213.0" width="693.0" x="30.0" y="0.0"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="sid-1402CFA5-62A0-4ADA-935A-05E5892D923D" id="sid-
    <omgdc:Bounds height="80.0" width="100.0" x="120.0" y="45.0"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="sid-6662DFDB-9ADE-4B79-B472-474B7B2B1271" id="sid-
    <omgdc:Bounds height="30.0" width="30.0" x="45.0" y="70.0"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape bpmnElement="sid-5334DCE4-C0B2-4F8F-B3EC-A6019C130D32" id="sid-
    <omgdc:Bounds height="40.0" width="40.0" x="255.0" y="65.0"/>
    </bpmndi:BPMNShape>
  </bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>

```

Figure 3.15.: BPMN2 XML file with diagram information

The BPMN2 standard including the BPMN2 XML format and execution semantics definition for

the execution of BPMN2 models is a major step in simplifying and unifying the work around modeling and execution of BPs. BPMN2 is one of the most widely-used BP modeling language nowadays and it is also the notation chosen in MINERVA framework, so it will be used throughout all chapters of this thesis.

### 3.1.3.2. XML Process Definition Language (XPDL)

XPDL [WfMC, 2008] is the (WfMC)<sup>4</sup> XML process definition language, which was first released as Workflow Process Definition Language (WPDL) at the beginning of the nineties for the enactment of BPs in workflows. After XML appeared it was released in its first XML version at the start of the third millennium. With the advent of BPMN, the XPDL standard was aligned to it to be the exchange format also for BPMN models, by integrating several elements as defined in the BPMN standard. Figure 3.16 displays the evolution of the XPDL standard, as compared to the BPMN standard.

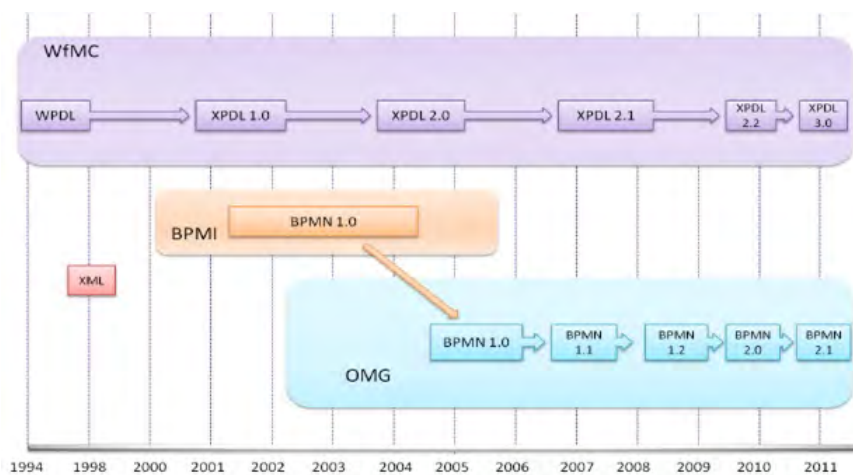


Figure 3.16.: XPDL standard evolution with BPMN adapted from [Shapiro and Gagne, 2010]

In the WfMC reference architecture, which describes the main pieces of a workflow management system at a high level, XPDL is the interface between process definition tools and the enactment service [Havey, 2005]. In Figure 3.17 the WfMC reference architecture is presented, where boxes represent components and arrows represent interfaces.

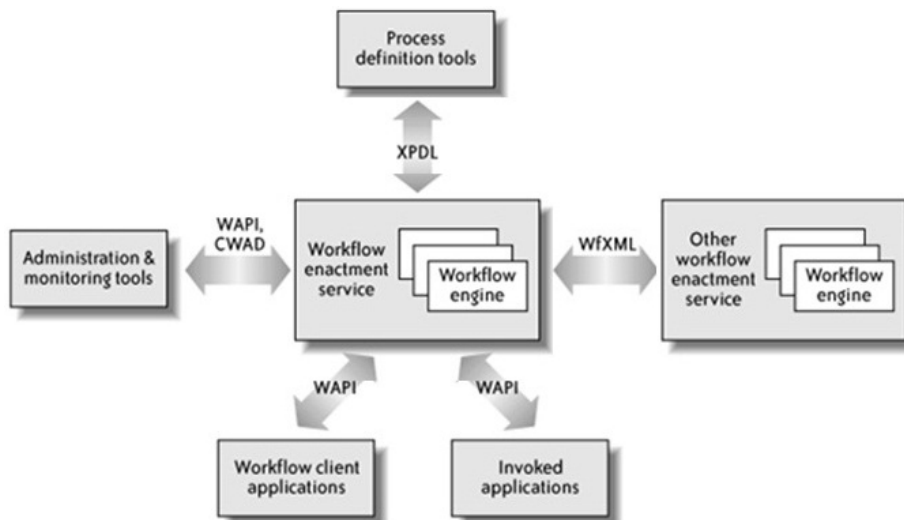


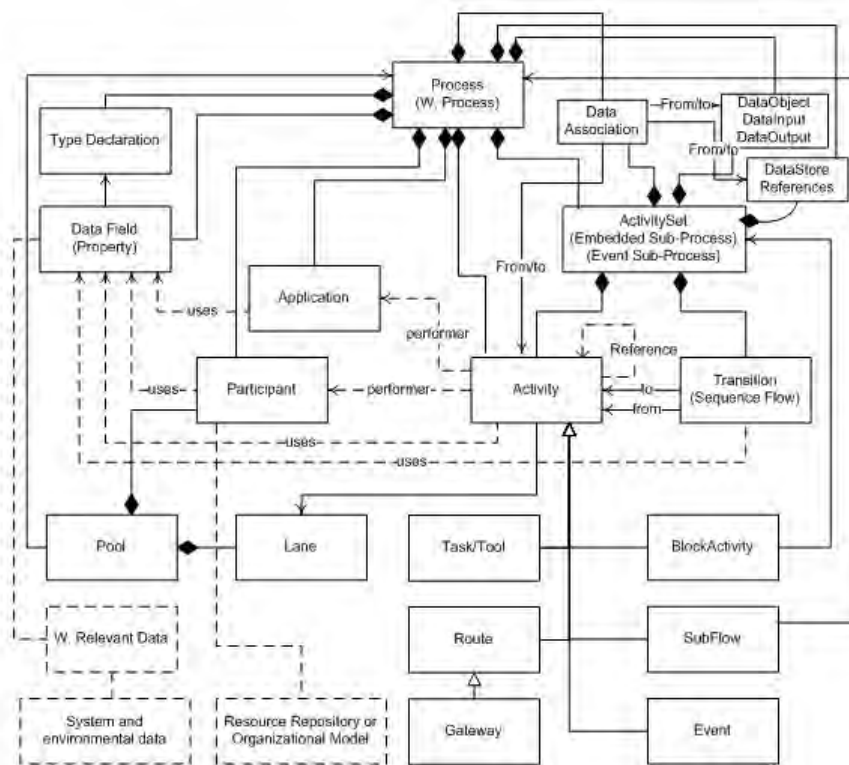
Figure 3.17.: WfMC reference architecture from [Havey, 2005]

<sup>4</sup><http://www.wfmc.org>

Process definition tools allow a BP to be modeled and exported into XPDL for its enactment in the workflow engine, XPDL being the corresponding interface. The enactment service (workflow engine) runs the process, creating and managing the execution of BP instances. It provides interfaces to external (client and invoked) applications, to other workflow engines, and to administration and monitoring tools which participate in the execution of the process [Havey, 2005].

A (workflow) process, is the central construct of XPDL, and consists of several elements that include activities and a set of transitions. It integrates BPMN elements such as Pool and Lane, Gateway, Event, and more recently in version 2.2 elements from BPMN2 such as DataObject, among others. The process definition provides the basis for the description of a process that can be used for the creation and control of instances during enactment, documentation and visualization.

“The process definition may contain references to subflows, separately defined, which make up part of the overall process definition. An initial process definition will contain at least the minimal set of objects and attributes necessary to initiate and support process execution. Some of these objects and attributes will be inherited by each created instance of the process” [WfMC, 2008]. In Figure 3.18 the process definition metamodel of version 2.2 is shown.



**Figure 3.18.:** XPDL process definition metamodel from [Shapiro and Gagne, 2010]

Figure 3.19 sets out an example of the structure of an XPDL file. It is worth noting that, in contrast to BPMN2, the graphical information for each element is integrated in the definition of the corresponding element, by means of a tag to define its representation. Note that it is an incomplete definition of a process just to illustrate an example, where several closing tags are missing.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://www.wfmc.org/2004/XPDL2.0alpha" xmlns:g360="http://www.global360.com/XPDL2.0alpha" Id="votingprocess"
xmlns:deprecated="http://www.wfmc.org/2002/XPDL1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Name="votingprocess"
xsi:schemaLocation="http://www.wfmc.org/2004/XPDL2.0alpha
C:\DOCUMENT~1\ROBERT~1\MYDOCU~1\vaapevisions\upm\schema\upm\xpdl_24.xsd">
  <PackageHeader>
    <XPDLVersion>2.0</XPDLVersion> <Vendor>Global 360</Vendor> <Created>10/14/2007 18:40:23 PM</Created>
  </PackageHeader>
  <Pools>
    <Pool Process="2" Id="86" Name="Voting Members" Orientation="VERTICAL" BoundaryVisible="true">
      <Lanes>
        <Lane Id="2" Name="Lane-2" ParentLane="86">
          <NodeGraphicsInfos>
            <NodeGraphicsInfo ToolId="G360_PMAE" IsVisible="true" Page="1" Height="46.0" Width="1089.0" LaneId="2">
              <Coordinates XCoordinate="75.0" YCoordinate="913.0"/>
            </NodeGraphicsInfo>
          </NodeGraphicsInfos>
        </Lane>
      </Lanes>
      <NodeGraphicsInfos>
        <NodeGraphicsInfo ToolId="G360_PMAE" IsVisible="true" Page="1" Height="76.0" Width="1075.0">
          <Coordinates XCoordinate="72.0" YCoordinate="886.0"/>
        </NodeGraphicsInfo>
      </NodeGraphicsInfos>
    </Pool>
    <MessageFlows>
      <MessageFlow Id="88" Name="Issue Announcement" Source="20" Target="86">
        <ConnectorGraphicsInfos>
          <ConnectorGraphicsInfo ToolId="G360_PMAE" IsVisible="true" Page="1">
            <Coordinates XCoordinate="268.0" YCoordinate="250.0"/>
            <Coordinates XCoordinate="271.0" YCoordinate="886.0"/>
            <g360:Font g360:Domain="Java" g360:Family="SansSerif" g360:Size="12" g360:Style="1"/>
            <g360:NameOffset>
              <Coordinates XCoordinate="45.0" YCoordinate="188.5"/>
            </g360:NameOffset>
          </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
      </MessageFlow>
    </MessageFlows>
    <Activities>
      <Activity Id="26" Name="Prepare Results" Status="None" StartQuantity="1" IsATransaction="false">
        <TransitionRestrictions>
          <TransitionRestriction>
            <Split Type="AND">
              <TransitionRefs>
                <TransitionRef Id="32"/>
                <TransitionRef Id="33"/>
              </TransitionRefs>
            </Split>
          </TransitionRestriction>
        </TransitionRestrictions>
        <NodeGraphicsInfos>
          <NodeGraphicsInfo ToolId="G360_PMAE" IsVisible="true" Page="1" Height="50.0" Width="75.0" LaneId="0">
            <Coordinates XCoordinate="620.0" YCoordinate="275.0"/>
          </NodeGraphicsInfo>
        </NodeGraphicsInfos>
      </Activity>
    </Activities>
  </Package>

```

Figure 3.19.: XPDl file example from [Wfmc, 2008]

XPDl is the defined enactment language of several process engines nowadays<sup>5</sup>. In this thesis it is used as one of the languages chosen for BP execution, which is obtained from a BPMN2 model.

### 3.1.3.3. Web Services BP Execution Language (WS-BPEL)

WS-BPEL [OASIS, 2007] from (OASIS)<sup>6</sup>, formerly BP Execution Language for Web Services (BPEL4WS) results from the integration of the languages WSFL from IBM, XLANG from Microsoft and PD4J from BEA Systems, its first version being launched at the beginning of this century. The change of name to WS-BPEL was made when it became an OASIS standard.

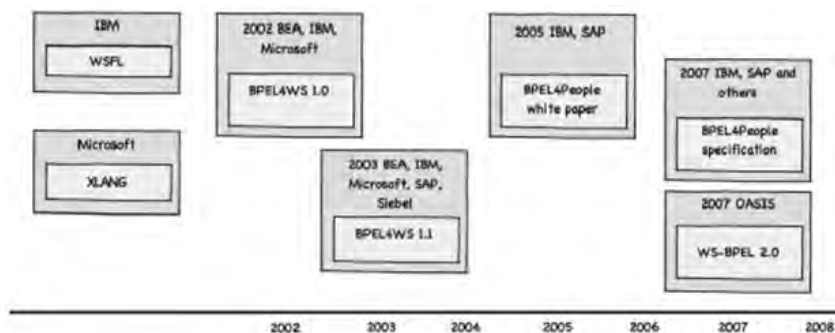


Figure 3.20.: WS-BPEL standard evolution from [Rademakers and van Liempd, 2011-2012]

<sup>5</sup><http://www.wfmc.org/xpdl-implementations.html>

<sup>6</sup><http://www.oasis-open.org>

A WS-BPEL process definition involves two types of files: the WS-BPEL file defining the process and the Web Services Definition Language (WSDL) files specifying the WS implemented by and called by the process [Havey, 2005], as shown in Figure 3.21.

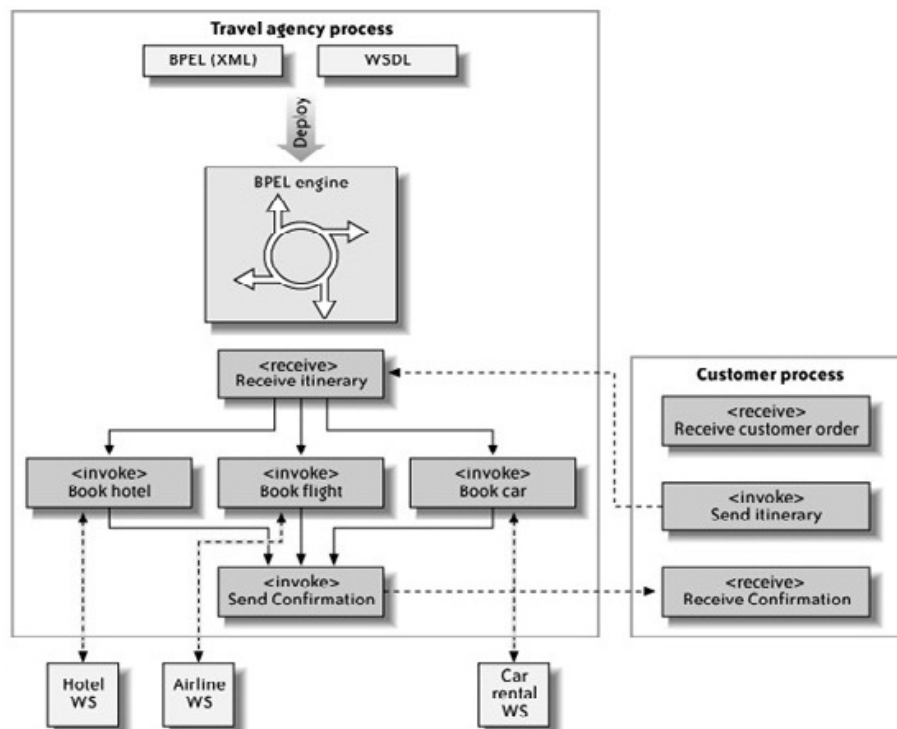


Figure 3.21.: WS-BPEL process execution from [Havey, 2005]

The WS-BPEL engine executes the process receiving invocations and invoking the corresponding WS as defined by the associated WSDL. In the example, the process receives an invocation from the customer process in the Receive itinerary activity and after that three WS are invoked from the Hotel, Airline and Car rental, and after receiving the corresponding answers the customer process is invoked to send the confirmation.

A WS-BPEL process is a container for the objects in the process, such as activities, partner links, correlation sets, variables, and handlers for compensation, faults, and events, where partner links represent message exchanges between two partners and are referenced by activities involving WS requests. The process is the central construction of WS-BPEL, to which the rest of the elements are related. Several types of activities are defined such as assign and validate, and three for WS: invoke, receive and reply, among several other constructions [Hornung et al., 2006].

A WS-BPEL metamodel is shown in Figure 3.22 and in Figure 3.23 an example of the structure of a WS-BPEL file is presented. In the first part the partner links are defined, the process variables and correlations set, and the process itself begins with the receive activity. Note that it is an incomplete definition of a process, where several closing tags are missing, and is given only by way of example.

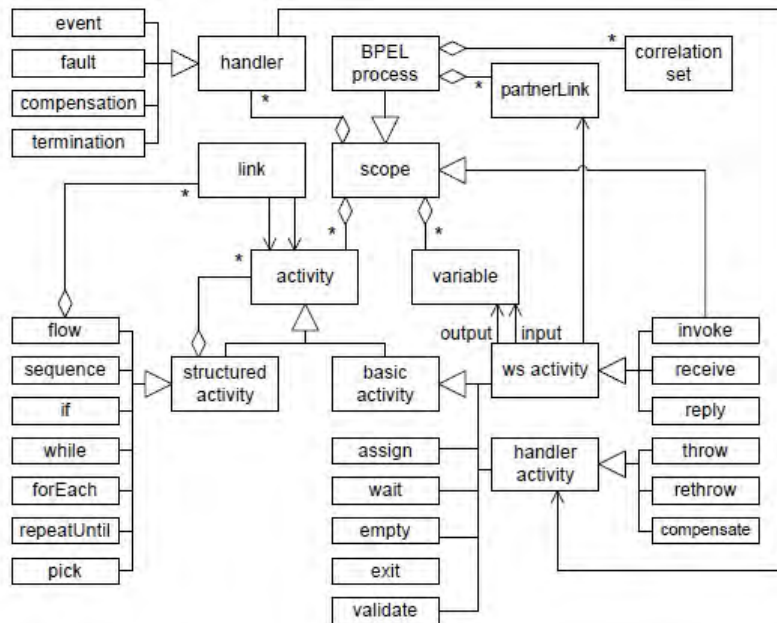


Figure 3.22.: WS-BPEL metamodel from [Hornung et al., 2006]

```

<process name="InsuranceClaim"
  targetNamespace="http://acm.org/samples" suppressJoinFailure="yes"
  xmlns:tns=http://acm.org/samples
  xmlns=http://schemas.xmlsoap.org/ws/2003/03/business-process/
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns:addressing=http://schemas.xmlsoap.org/ws/2003/03/addressing
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
  <partnerLinks>
    <partnerLink name="client" partnerLinkType="tns:InsuranceClaim"
      myRole="InsuranceClaimProvider"/>
    <partnerLink name="worklist" partnerLinkType="task:TaskManager"
      partnerRole="TaskManager" myRole="TaskManagerRequester"/>
  </partnerLinks>
  <variables>
    <variable name="status" type="xsd:string"/>
    <variable name="initiateMsg" messageType="tns:InsuranceClaimMsg"/>
    <variable name="killEv" messageType="tns:InsuranceClaimMsg"/>
    <variable name="taskResponse" messageType="task:taskMessage"/>
  </variables>
  <correlationSets>
    <correlationSet name="claim" properties="tns:claimID"/>
  </correlationSets>
  <receive partnerLink="client" portType="tns:InsuranceClaim"
    operation="initiate" variable="initiateMsg" createInstance="yes"
    name="initiateEvent">
    <correlations>
      <correlation set="claim" initiate="yes"/>
    </correlations>
  </receive>
  <invoke name="evalClaim" partnerLink="worklist" portType="task:TaskManager"
    operation="evalClaim" inputVariable="initiateMsg"/>
  <pick name="analyzePick">
    <onMessage partnerLink="worklist" portType="task:TaskManagerCallback"
      operation="onTaskResult" variable="taskResponse">
      <!-- From response extract status and set to variable 'status' -->
      <assign name="setStatus">
        <copy>
          <from variable="taskResponse" part="payload"
            query="/tns:taskMessage/tns:result="/>
          <to variable="status"/>
        </copy>
      </assign>
    </onMessage>
  </pick>

```

Figure 3.23.: WS-BPEL example file from [Havey, 2005]

WS-BPEL allows only automatic interaction via WS, so to integrate human interactions such as the ones XPDL and BPMN2 provide, two standards for extending WS-BPEL were defined: the WS-HumanTask [OASIS, 2010b] defines human tasks including their behavior, properties and operations to manipulate them, the BPEL4People [OASIS, 2010a] extends WS-BPEL to address human interactions, defining a new type of basic activity which is implemented as a human task, based on the WS-HumanTask specification.

WS-BPEL is the defined and execution language of several process engines nowadays<sup>7</sup>. In this thesis it is used as one of the languages choosen for BP execution, which is obtained from a BPMN2 model.

## 3.2. Service Oriented Computing (SOC)

This section presents the main elements involved in the Service Oriented Computing (SOC) paradigm, to develop software systems based on the definition, design, implementation and execution of services. In the first place concepts and definitions for SOC and its related terms, Service Oriented Development (SOD) and Service Oriented Architecture (SOA) , are presented, secondly service concepts are introduced including service design principles, and finally a key standard for service modeling, SoaML [OMG, 2009b] is described.

### 3.2.1. SOC, SOE, SOD and SOA

Recent years have seen major changes in the area of computing, in the proliferation of new technologies, methodologies and development approaches that have influenced today's organizations, while, at the same time and on the other hand, changes in the requirements and needs in organizations have affected the way of developing and executing software. The explosion of the use of Internet by organizations, presents several advantages and challenges for the way they conduct their business, as well as the way they implement their processes and interact with other organizations.

The Information Technology (IT) area in today's organizations can be characterized as supporting diverse heterogeneous systems with complex dependencies, which have grown separately and mingled over the years. A challenge that arises is to integrate them so they can react nimbly to changing business requirements, mainly in two aspects: the BPs in the organization and the technologies available. The definition and availability of services to support this vision for the entire organization is the basis of service orientation [Krafzig et al., 2005].

In this thesis "Service-oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications" [Papazoglou and Georgakopoulos, 2003]. SOC uses services "to support the development of rapid, low-cost, interoperable, evolvable, and massively distributed applications" Papazoglou et al. [2007]. The service-oriented approach is based on the idea of composing applications by discovering and invoking network-available services to provide support to the realization of some task, which is independent of specific programming languages or operating systems Papazoglou et al. [2007].

"Key to realizing this vision is the service-oriented architecture (SOA). SOA is a logical way of designing a software system to provide services either to end-user applications or other services distributed in a network, via published and discoverable interfaces" [Papazoglou et al., 2007]. In [Erl, 2005] "Service-oriented architecture is a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic. Individually, these units can be distributed.. SOA encourages individual units of logic to exist autonomously yet not isolated from each other.. these units of logic are known as services".

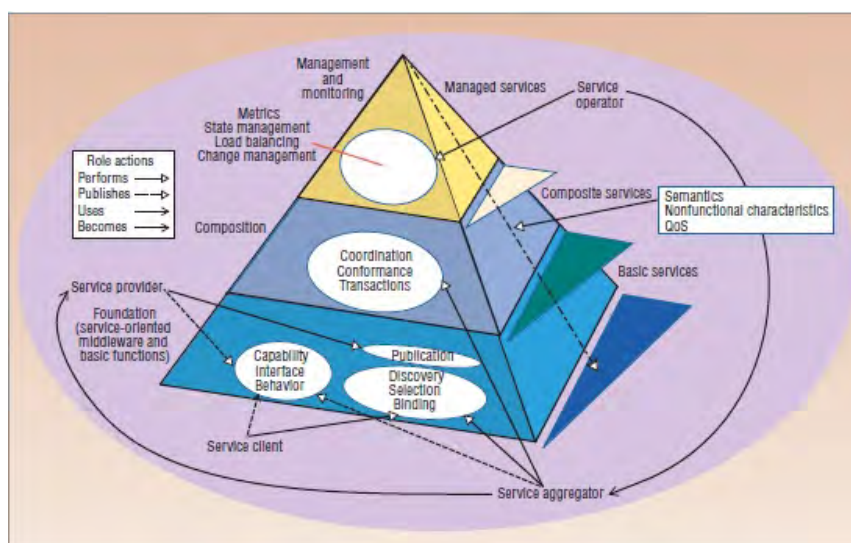
For [Krafzig et al., 2005] SOA is an architecture style of reusable software-based services, with well-defined public interfaces, where suppliers and consumers of services interact in a decoupled

---

<sup>7</sup><http://bpel.xml.org/products>

way to conduct business processes. A service provides business logic and data, a service contract, restrictions for the consumer, and interfaces that expose functionality. This is the definition we adopt in this thesis, where an architectural style “determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined” [Garlan and Shaw, 1994]. SOAs are realized mainly by means of Web Services (WS).

In [Papazoglou and Georgakopoulos, 2003] an extended SOA is defined thus “SOC involves the service layers, functionality, and roles as described by the extended service-oriented architecture (SOA). Basic services, their descriptions, and basic operations (publication, discovery, selection, and binding) that produce or utilize such descriptions constitute the SOA foundation. The higher layers in the SOA pyramid provide additional support required for service composition and service management”. The extended SOA separates functionality into three planes: service foundations at the bottom (basic services), service composition in the middle (composite services), and service management and monitoring on top (managed services), as shown in Figure 3.24, to create a reactive and adaptive IT environment [Papazoglou et al., 2007].

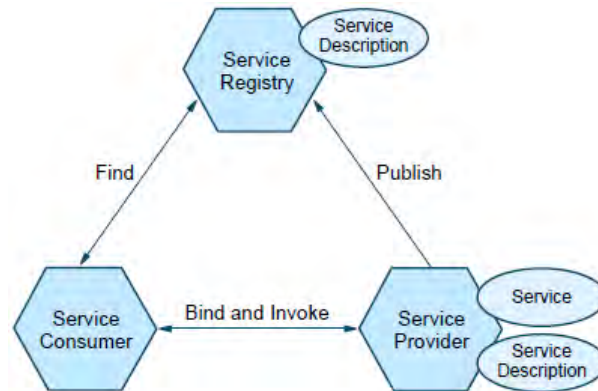


**Figure 3.24.:** Extended SOA layers from [Papazoglou et al., 2007]

SOA includes other elements such as service registry and service bus. A service registry provides facilities to publish and discover services and to get the information to use them (i.e. UDDI) such as physical location, provider and contact information, user fees, technical restrictions, security aspects and SLAs (Service Level Agreements) for the service Krafzig et al. [2005]. A service bus connects participants in a SOA by providing: connectivity between SOA elements, use of different technologies and products, heterogeneity of communication concepts such as synchronous and asynchronous communication, and technical services such as security, message transformation and auditing [Krafzig et al., 2005].

The collaboration model between participants in a SOA follows the find-bind-invoke paradigm which is shown in Figure 3.25. The find-bind-invoke paradigm allows services to be decoupled from each other and works in this way: the service provider owns the implementation and description of the service and registers this description in a service registry for the service to be accessible for consumers; a service consumer searches in the registry for a service provider fulfilling its needs and when successful the registry returns the service provider physical information for the service consumer to actually invoke the service, which is done by the binding and invoke indication from service consumer to service provider. When a registry is not available or used, the service consumer must know the physical address of the service to be invoked at design time, to perform the binding.





**Figure 3.25.:** Find-bind-invoke paradigm in SOA from [Endrei et al., 2004]

“Service Oriented Engineering (SOE), i.e. service-oriented analysis, design and development techniques and methodologies, are crucial elements for developing meaningful services and business process specifications and are an important requirement for SOA applications that leverage Web services. Service-oriented engineering activities help develop meaningful services, service compositions and techniques for managing services, and apply to the three service planes defined in the extended SOA” [Papazoglou et al., 2007].

“Service engineering must handle the complete service lifecycle covering both the provider and consumer perspectives, addressing issues of service management and support for architectural frameworks. Service engineering is impractical without the existence of sound and methodical practices for service identification, modeling design, and implementation” [Karakostas and Zorgios, 2008].

Service Oriented Development (SOD) or “Service design and development is about identifying the right services, organizing them in a manageable hierarchy of composite services (smaller grained often supporting larger grained), choreographing them together for supporting a business process” [Papazoglou and van den Heuvel, 2006]. SOD refers to the development of software based on services while SOE covers the complete services lifecycle, although this separation is somewhat unclear and sometimes both terms are used interchangeably. In Figure 3.26 the relationships between SOC, SOE, SOD and SOA are shown in diagram form.



**Figure 3.26.:** SOC, SOE, SOD and SOA relationships

### 3.2.2. Service Concepts

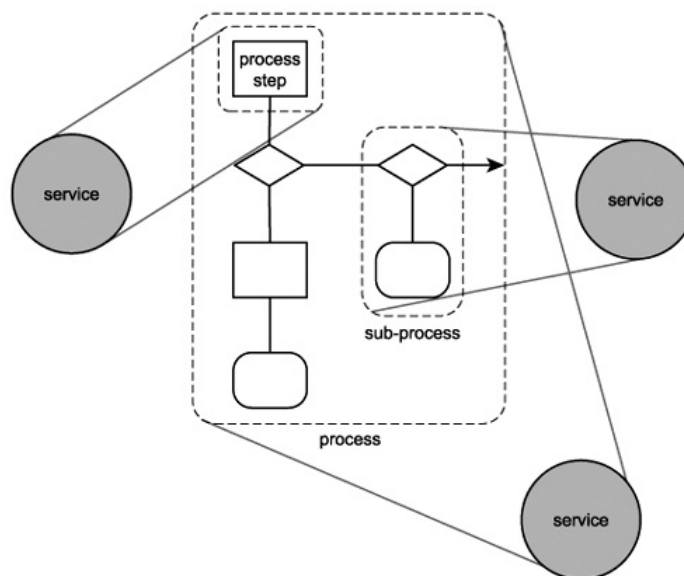
In this section what a service means is discussed in the first place, describing several related concepts such as communication patterns between services and services classifications. After these, service design principles that have to be taken into account for modelling services are presented.

### 3.2.2.1. Service definitions

Services are the key elements in service oriented approaches. In [Papazoglou et al., 2007] “services are autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways .. a business service or process can be composed of finer-grained (atomic) services that need to be supported by infrastructure services and management services, such as those providing technical utility like logging, security or authentication, and those that manage resources”.

In [Krafzig et al., 2005] “a service provides business logic and data, a service contract, restrictions for the consumer, and interfaces that expose the functionality”. The technical realization of the service fulfills the defined contract, in the form of components, programs, configuration data and data bases. Different types of services can be defined which present different characteristics, for example process-oriented services realize BPs which are composed of sequences of invocations to defined services.

[Erl, 2005] states that services are independent logic units, which encapsulate logic in a context that can be specific to a business task, a business entity or any other logical grouping. The size and scope of the logic that a service represents can vary, and the logic of a service can contain logic from another services in which case one or more services are composed in another. Each service can encapsulate an individual step, a sub-process containing several steps, or a complete BP, as shown in Figure 3.27.



**Figure 3.27.:** Services logic encapsulation from [Erl, 2005]

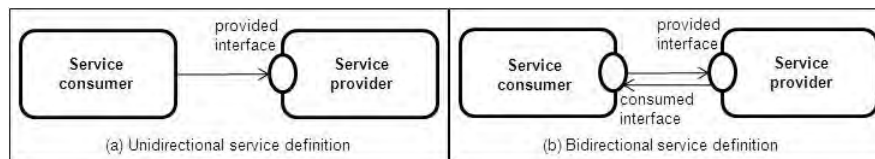
In [OMG, 2009b] “a service is defined as the delivery of value to another party, enabled by one or more capabilities. Here, the access to the service is provided using a prescribed contract and is exercised consistent with constraints and policies as specified by the service contract. A service is provided by a participant acting as the provider of the service—for use by others. The eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider”, which is based on [OASIS, 2006].

In [OpenGroup, 2008] a service is “a logical representation of a repeatable business activity that has a specified outcome (e.g. check customer credit). It is self-contained, may be composed of other services, and is a "black box" to its consumers. A service has a provider, may have multiple consumers, and produces one or more effects (which have value to the consumers)”; which is the one adopted in this thesis, adding the elements from [Krafzig et al., 2005][OMG, 2009b]: service contract, restrictions for the consumer, and interfaces that expose the functionality, the vision

of scope from [Erl, 2005] and the vision of autonomous and platform-independent entities by [Papazoglou et al., 2007].

The interaction with a service can be described by means of Message exchange patterns (MEPs) [Erl, 2005] which are applied synchronously or asynchronously. Basic patterns are the Fire-and-forget which is based on the unidirectional transmission of messages from a source to one or more destinations, and the Request-response which establishes a simple exchange between a source and a destination, in which a message is first transmitted from a source to the destination and upon receiving the message the destination responds with a message back to the source [Erl, 2005].

[OMG, 2009b] describes basic communication patterns with a service as being unidirectional or bidirectional, referring to the cases in which there is only a simple provided interface, or where there is a provided and a consumed interface, respectively. In Figure 3.28 these communication patterns are shown (a) unidirectional and (b) bidirectional.

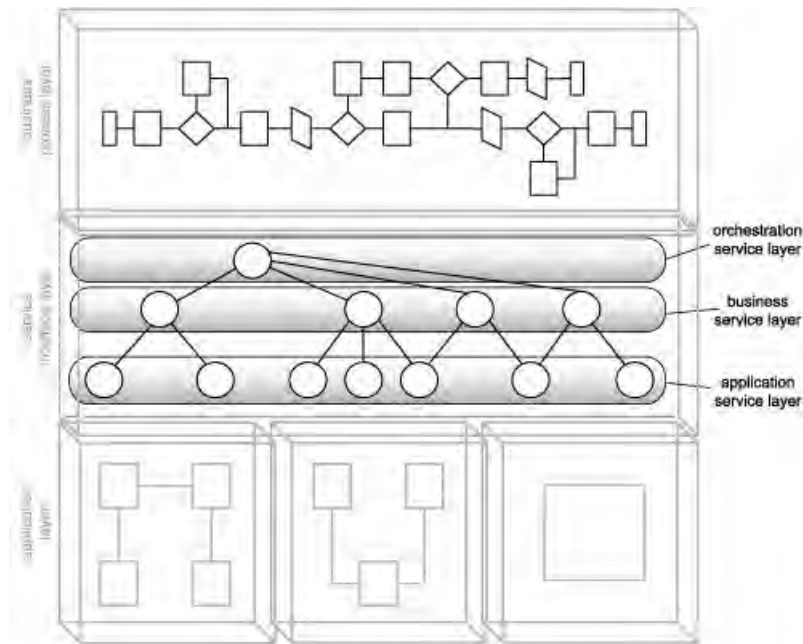


**Figure 3.28.:** (a) unidirectional and (b) bidirectional service definition based on interfaces

Services can be classified in hierarchies to organize the services layer and to provide a conceptual context for the identification of services, which will help in the design process. Several classifications exist from of which we present two, briefly, to illustrate the vision of classification of services, which are somewhat similar between proposals. The main idea is to have a hierarchy of service layers in which the upper layer corresponds to the orchestration of services (the representation of the BP) and below it different layers represent business and technological services in a different granularity, to be orchestrated to realize the BP.

In [Krafzig et al., 2005] services are classified in basic services, intermediate services, process-oriented services and public enterprise services. Basic services are further broken down into data-centric services and logic-centric services, where the first provide access to data encapsulating manipulation of data for a single business entity each service, and the second to business rules. Intermediate services provide bridges to ease technical inconsistencies or conceptual discrepancies, and they are broken down into gateways, adapters, facades and functionality-adding. Gateways provide technological bridges by representing underlying functionality in another technology, adapters provide a new vision over underlying functionality, facade can aggregate several services to provide a single vision and functionality-adding allows to add functionality to an existing service without changing it. Process-oriented services represent the orchestration of services by means of defined sequences of invocations to other services, in an orchestration (process) engine. Public enterprise services are services offered to clients and external partners, adding, for example, user fees and security.

In [Erl, 2005] services are classified in application services, business services and process services. Application services provide reusable functions related to processing data within new or legacy application environments; this is to express technology-specific functionality. They are further broken down into utility services to provide reusable operations and wrappers to encapsulate legacy systems. Business services represent business logic in its pure form, and can also be made up of other services to provide the required functionality. They are further broken down into task-centric services which encapsulate business logic specific to a task or BP, and entity-centric services which encapsulate a specific business entity (such as an invoice). Process services are composed of business and application services according to the business rules and business logic embedded in the process definition, and correspond to the implementation of the BPs in orchestration (process) engines. In Figure 3.29 this latter classification is shown to illustrate the concept.



**Figure 3.29.:** Services classification according to [Erl, 2005]

In MINERVA the classification of services is included in BPSOM in a Design Discipline activity, which is presented in chapter 7.

### 3.2.2.2. Service design principles

The modelling of services should follow well-known design principles [Papazoglou and van den Heuvel, 2007] such as coupling and cohesion, and the definition of the granularity of the service, which although is not a design principle is related to the first two. The main objective is as with traditional design a low coupling between different services and a high cohesion in each service, which will also lead to a well-defined level of granularity for the services designed.

Service coupling refers to the degree of interdependence between BPs or services, as design coupling is traditionally defined, where the objective is to minimize coupling. BPs should be as self-contained as possible ensuring they do not need knowledge or service of other BPs. Orchestrated services within a process should be loosely coupled to each other, avoiding interdependencies at the level of data or control logic [Papazoglou and van den Heuvel, 2007]. Three ways of coupling are defined: representational, identity and communication protocol coupling:

- **representational coupling:** a BP should not depend on specific representational or implementation details of the languages used to compose their underlying services. It supports interchangeable/replaceable services that can be swapped with new service implementations (or new ones from another provider) without affecting the BP functionality, and multiple service versions to support parts of a BP depending on the application's needs.
- **Identity coupling:** connection channels between services should not know who provides the service, especially when they are likely to change or when discovering the best service provider is not unimportant.
- **Communication protocol coupling:** minimize the number of messages exchanged between a sender and a destination, such as one-way, request/response, or solicit/response.

Service cohesion [Papazoglou and van den Heuvel, 2007] refers to the degree of strength of functional relatedness between operations in a service or BP, as design cohesion is defined, where the objective is to maximize it. BPs should be cohesive and autonomous and the services realizing them and its defined operations should be strongly and genuinely related to one another. To increase cohesion three types are defined: functional service cohesion, communicational service cohesion and logical service cohesion.

- Functional service cohesion: such a BP should perform only one problem-related task and be realized only by services necessary for that purpose.
- Communicational service cohesion: such a BP is one whose activities and services use the same set of input and output messages, and that is why it is decoupled from other processes.
- Logical service cohesion: this kind of BP performs a set of independent but logically similar functions representing alternatives tied together by the control flow, such as mode of payment.

Service granularity refers to the scope of functionality exposed by a service, it can be fine-grained providing a small amount of BP usefulness, such as basic data access, or can be coarse-grained providing the complete processing for a given service. In the first case, the service might present separate operations, for example to create an order, add an item, and so forth, where the number of messages needed results in increased network traffic, which might be acceptable for Enterprise Application Integration (EAI) applications within one organization. For interactions outside the organization it would be preferable to design coarse-grained services providing access to a complete BP to accomplish a specific business task, which can be created from one or more existing systems by defining and exposing interfaces that meet business process requirements and by composing several services [Papazoglou and van den Heuvel, 2006].

However in [Erl, 2005] a discussion is provided on the suitable level of service granularity, stating that “the coarser the granularity of an interface, the less reuse it may be able to offer. If multiple functions are bundled into a single operation, it may be undesirable for requestors who only require the use of one of those functions. Additionally, some coarse-grained interfaces may actually impose redundant processing or data exchange by forcing requestors to submit data not relevant to a particular activity”. It is clear that the level of granularity will depend on the context and technical possibilities of each organization.

Another key design principle presented in [Erl, 2005] corresponds to the design for change principle, under the name of design service operations for extensibility. When change arises and depending on the type of BP change, it could result in the need to broaden the scope of entities, so services may need to be extended. It is worth looking first for the possibility of composing existing services, and if this is not possible, decide on whether there is a need to extend the interface or only the underlying implementation, to add the change needed. [Erl, 2005] also recommends incorporating in service design a speculative analysis of how that service may be utilized outside its initial application boundaries, identifying future service requestors and integrating their anticipated requirements into the current service design, to cover many other possible scenarios.

There are not many specific service modelling notations, and UML was mainly used to describe service design models by means of subsystems, classes and components. With the advent of SoaML which is an UML profile and metamodel, it is rapidly becoming the preferred notation. This corresponds to the modeling of services at a pure design level with no information about the implementation of services, on the other hand, when implementation details are known such as the technology or platform to be used, the description of services can include the specification of Web Services (WS) by means of defining their WSDL (Web Service Description Language)<sup>8</sup>, and the composition of modules for implementation and assembly by means of the Service Component Architecture (SCA)<sup>9</sup>.

### 3.2.3. Main standards for service orientation

In this section the main standard related to services modeling is presented, the Service Oriented Architecture Modeling Language (SoaML) as defined by OMG.

---

<sup>8</sup><http://www.w3.org/TR/wsdl20/>

<sup>9</sup><http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>

### 3.2.3.1. Service Oriented Architecture Modeling (SoaML)

SoaML [OMG, 2009b] was released in its first version beta 1 [OMG, 2009c] in April 2009, and updated by the current version beta 2 on December 2009. It provides a metamodel and a UML profile for the specification and design of services within a service-oriented architecture.

The goals of SoaML are to support the activities of service modeling and design and to fit into an overall model-driven development approach [OMG, 2009b]. It defines specific stereotypes to be used when modeling services within a SOA but with no reference to implementation details. From this design, model code can be obtained either manually or automatically, as the elements in SoaML allow all the information needed for implementing services to be specified. SoaML defines the following main elements to model services:

- **ServiceArchitecture:** is a network of participant roles providing and consuming services to fulfill a purpose. The services architecture defines the requirements for the types of participants and service realizations that fulfill those roles [OMG, 2009b]. It allows a high level view of the participants to be shown, as well as the services needed for the participants to be able to interact with each other, and the role each participant plays within each service (as provider or consumer).
- **Participant:** participants are either specific entities or kinds of entities that provide or use services. Participants can represent people, organizations or information system components. Participants may provide any number of services and may consume any number of services [OMG, 2009b].
- **Ports:** participants provide or consume services via ports. A port is the part or feature of a participant that is the interaction point for a service, where it is provided or consumed. A port where a service is offered corresponds to a «Service» port and the port where a service is consumed corresponds to a «Request» port [OMG, 2009b].
- **Service specification:** the description of how the participant interacts to provide or use a service is encapsulated in a specification for the service including its Interfaces and ServiceContract. When Interfaces are defined as ServiceInterfaces the ServiceContract might not be defined as the information can be detailed in them. ServiceInterfaces define a bidirectional way of communication with the service, while with UML simple Interfaces the communication can be defined as unidirectional or bidirectional, depending on the service definition.
- **Capabilities:** participants that provide a service must have a capability to provide it, they can be seen from two perspectives, capabilities that a participant has that can be exploited to provide services, and capabilities that an enterprise needs that can be used to identify candidate services.

In Figure 3.30 some SoaML elements are presented: (a) ServicesArchitecture showing participants, services and roles and (b) participants showing Service and Request ports and associated interfaces. In Figure 3.31 (a) a ServiceContract is presented showing the consumer and provider roles and the associated interfaces as their types, (b) the UML simple Interfaces definition for the services showing the dependencies defined with each other, and (c) a ServiceInterface definition of services, showing the realization of the interface from the provider and the use of the interface from the consumer.

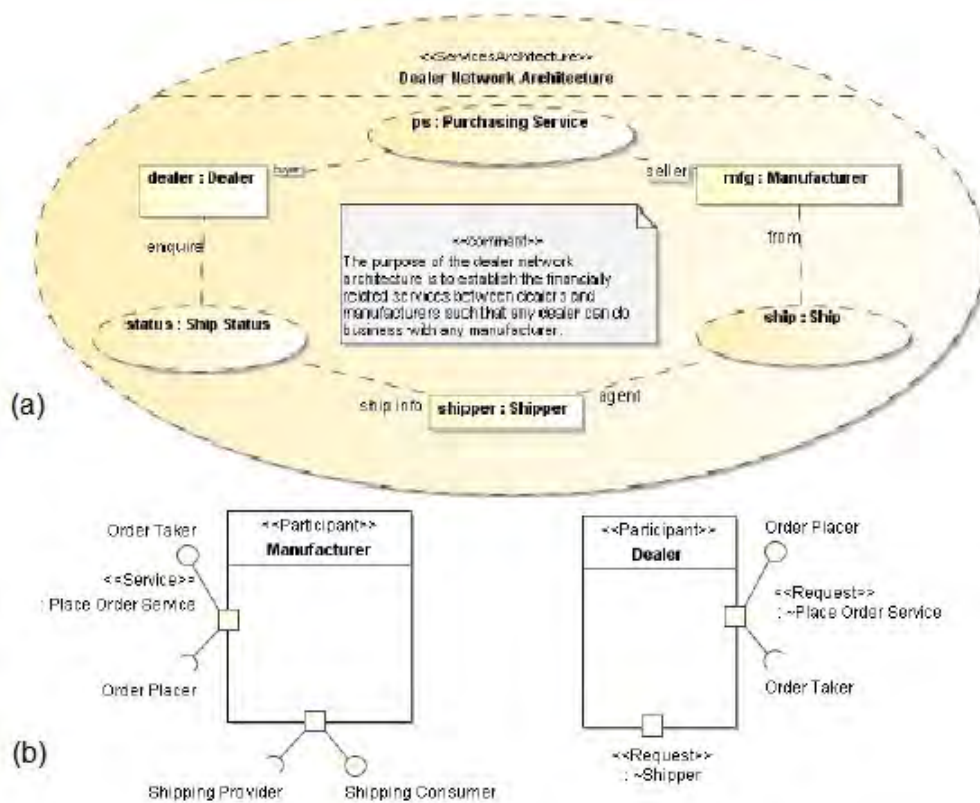


Figure 3.30.: SoaML ServicesArchitecture and Participants specification from [OMG, 2009b]

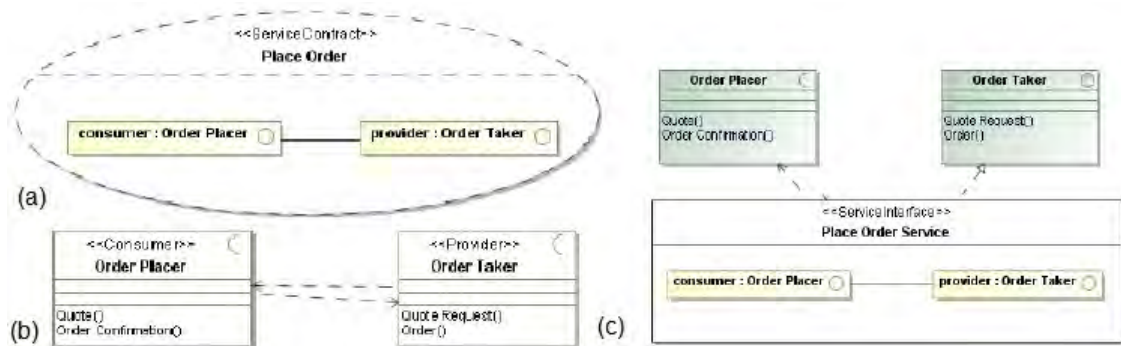


Figure 3.31.: SoaML ServiceContract, Interfaces and ServiceInterfaces specification from [OMG, 2009b]

The UML profile provides UML2 tools to be able to develop, transform and exchange services metamodels in a standard way, and also lays the foundation for new tools to extend UML2 to support services modeling in a first class way (see subsection 3.3.3 for UML profile vs. DSL definitions), providing also a mapping between the profile and the extended metamodel elements. In Figure 3.32 some stereotypes defined in the SoaML profile are shown.

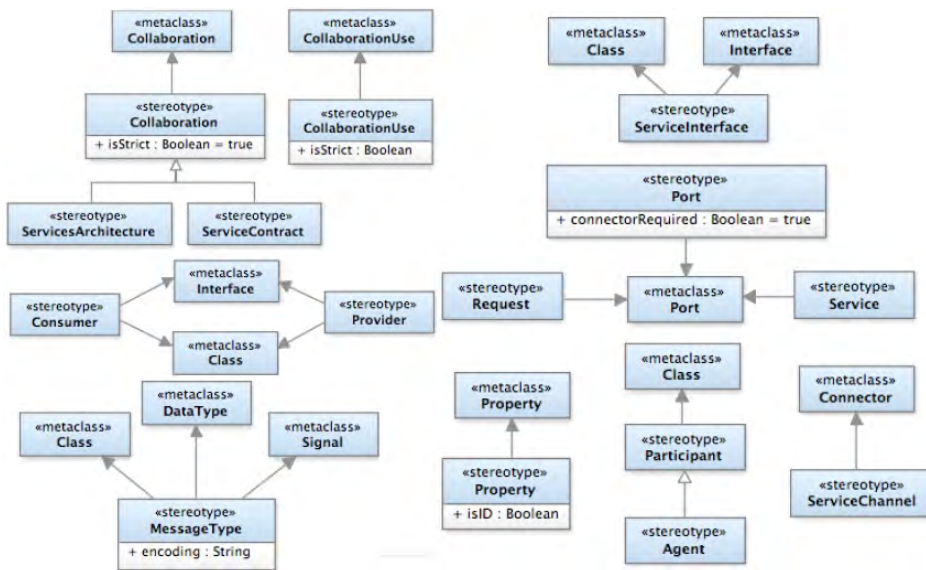


Figure 3.32.: Some stereotypes defined in the SoAML profile from [OMG, 2009b]

As presented previously the ServicesArchitecture, being an UML Collaboration, allows the participants, services (as ServiceContract CollaborationUse) and roles played by the participants within each service to be modeled. A ServiceContract is also a UML Collaboration defining the roles (consumer, provider applied on UML Class or Interface) and the Interface types for these roles. Service and Request are applied on UML Ports which are assigned to participants, whose stereotype is applied on a UML Class. A ServiceChannel is applied on a UML Connector, and a MessageType can be applied on a UML Class, DataType or Signal. In Figure 3.32 the SoAML metamodel elements are shown.

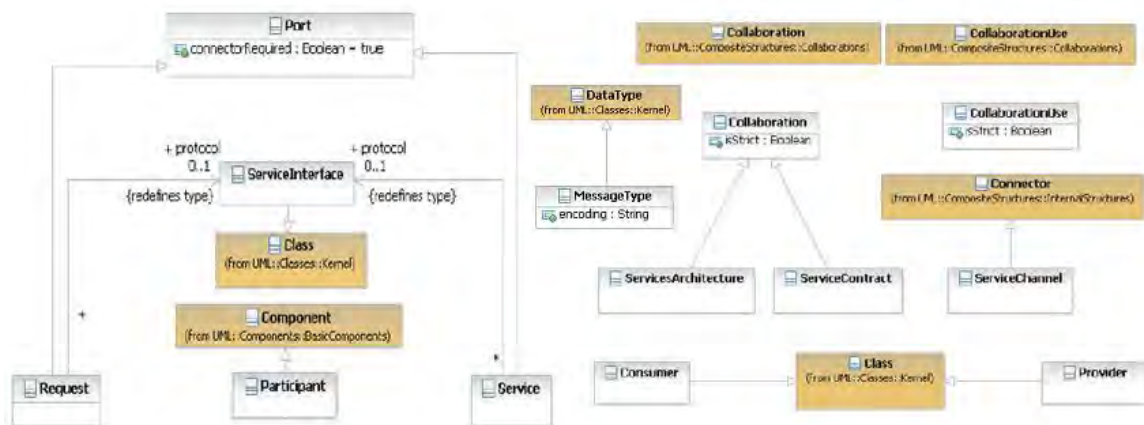


Figure 3.33.: Some elements defined in the SoAML metamodel from [OMG, 2009b]

SoAML is being rapidly adopted to model services based on UML, and it is also the selected notation in MINERVA framework, so it will be used throughout all chapters of this thesis. Both the SoAML metamodel and the SoAML profile are used in MINERVA, the first one to be able to define the QVT transformations from BPMN2 models, and the second one to visualize the generated SoAML models with the SoAML profile application in the Eclipse SoAML plug-in we have developed, which are described in chapter 8 and chapter 9 respectively.



### 3.3. Model Driven Development (MDD)

This section presents the main elements that the Model Driven Development (MDD) paradigm comprises, to drive software development by models and for the automatic generation of artifacts from them. In the first place concepts and definitions for MDD and its related terms, Model Driven Engineering (MDE) and Model Driven Architecture (MDA), are presented, secondly models, metamodels and transformations definitions and their relationships are described, after that mechanisms for defining modeling languages in this context are presented and discussed, and finally three main standards for model-driven are presented: MDA [OMG, 2003], MOF [OMG, 2011b] and QVT [OMG, 2008c].

#### 3.3.1. MDE, MDD and MDA

During the past two decades important advances have been made in languages and platforms providing developers with a high level of abstraction for implementing software systems, although platform complexity has also grown faster than the ability of languages to mask it [Schmidt, 2006]. Developers nowadays face more complex and large systems, having to integrate several technologies and platforms, and as stated in [Schmidt, 2006] “.. since these platforms often evolve rapidly—and new platforms appear regularly— developers expend considerable effort manually porting application code to different platforms or newer versions of the same platform .. most application and platform code is still written and maintained manually using third-generation languages, which incurs excessive time and effort ..”

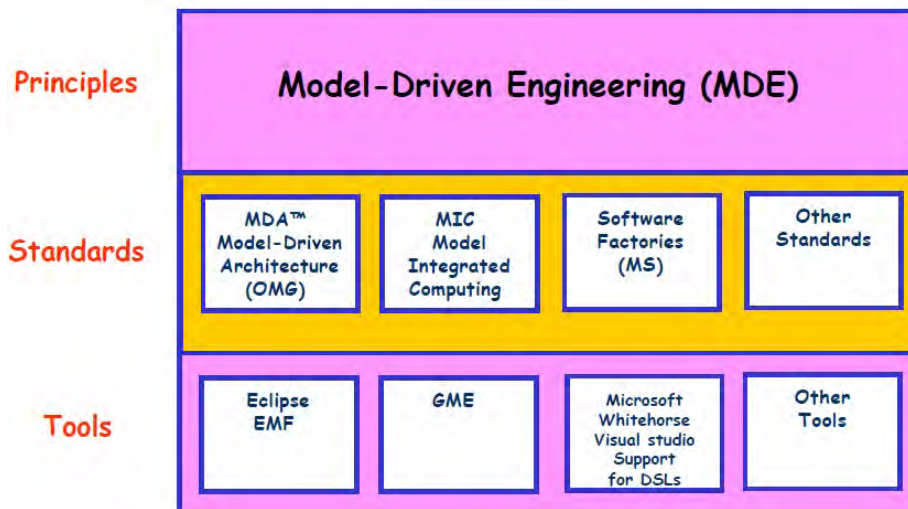
A change of paradigm from “code centric” to “model centric” is needed, to be able to significantly raise the level of abstraction [Vallecillo, 2000-2011] alleviating the described problems, but as stated in [Mellor et al., 2003] models are still seen by developers as “simply drawing pictures removed from real systems development”, and used merely with documentation purposes for implementation and rarely updated. When we look to other more mature engineering disciplines (building bridges, highways, ships, etc.) the use of models is a key element, as models help in: specifying the system (structure, behavior), communicating with other stakeholders, thinking about and validating the system (finding design errors and omissions, prototype the model, analyze properties), guiding the implementation of the system and understanding it if the system already exists [Vallecillo, 2000-2011].

Model Driven Engineering (MDE) refers to the systematic use of models as primary engineering artifacts throughout the engineering life cycle [Karakostas and Zorgios, 2008], encompassing all engineering efforts such as development, maintenance or reverse engineering, where a systematic approach implies a methodical approach that is repeatable and learnable through a step by step procedure. The main objective is to raise the level of abstraction “to address platform complexity and the inability of third-generation languages to alleviate this complexity and express domain concepts effectively”, and can be seen as an evolution of CASE tools [Schmidt, 2006]. As stated in [Schmidt, 2006] this is achievable by means of MDE technologies providing:

- Domain Specific Modeling Languages (DSML or DSL) “whose type systems formalize the application structure, behavior, and requirements within particular domains, such as .. online financial services, warehouse management, or even the domain of middleware platforms .. described using metamodels, which define the relationships among concepts in a domain and precisely specify the key semantics and constraints associated with these domain concepts”.
- “Transformation engines and generators that analyze certain aspects of models and then synthesize various types of artifacts, such as source code, simulation inputs, XML deployment descriptions, or alternative model representations .. this automated transformation process is often referred to as “correct-by-construction,” as opposed to conventional handcrafted “construct-by-correction” software development processes that are tedious and error prone”.

In [Bézivin, 2005] several objectives for MDE are stated: “separation from business-neutral descriptions and platform dependent implementations, the identification, precise expression, separation and combination of specific aspects of a system under development with domain-specific languages,

the establishment of precise relations between these different languages in a global framework and in particular the possibility to express operational transformations between them”. The scope of MDE is also described in [Bézivin, 2004] related to standards and tools as shown in Figure 3.34.



**Figure 3.34.:** MDE scope, standards and tools from [Bézivin, 2004]

Starting from the right of Figure 3.34 Software Factories from Microsoft are defined as “a model-driven product line - a product line automated by metadata captured by models using domain specific modeling languages” [Greenfield, 2003]. Conceptually they can be seen as the industrialization of software development “from craftsmanship to manufacturing” to support the product-line engineering process. It consists of an IDE specifically configured for the development of a specific kind of application, i.e. in a specific domain, making use of Domain Specific Modeling (DSM) by means of DSLs, frameworks and patterns, with its own tools MS/DSL providing facilities to manipulate them [Stahl and Volter, 2006].

Model Integrated Computing (MIC) uses domain-specific (modeling) languages (DSLs) to represent system elements and their relationships as well as their transformations to platform-specific artifacts [Balasubramanian et al., 2006]. It started in the context of distributed real time and embedded systems (DRE systems) and one of its main supporters is Vanderbilt University’s Institute for Software Integrated Systems (ISIS). Models are the center of the complete lifecycle of systems, rather than just for development, and the verification of models is a primary concern [Stahl and Volter, 2006]. The Generic Modeling Environment (GME) open source tool is provided, which is a metaprogrammable, domain-specific design environment that developers use both to create DSLs and models that conform to these DSLs within the same graphical environment, and to generate the associated artifacts [Balasubramanian et al., 2006].

Model Driven development (MDD) bases the development of software on models, automating the transformation of models from one form to another. Models are expressed in a language that exists at some (language) abstraction level, while the modeling language’s syntax and semantics are defined in a model of the modeling language, a metamodel. Expert knowledge is captured as mapping functions that transform one model to another [Mellor et al., 2003]. MDD implies the (semi) automated generation of implementation(s) from models, where models conform to metamodels, modeling languages are key elements and model transformation languages are also modeling languages [Vallecillo, 2000-2011], defining a general approach for development.

Model Driven Architecture (MDA) [OMG, 2003] is the OMG proposal for MDD, based on OMG standards such as Meta Object Facility (MOF) [OMG, 2011b], UML, Object Constraint Language (OCL), XMI and Query/View/Transformations (QVT) [OMG, 2008c], where MOF and UML allow the definition of new families of languages [Vallecillo, 2000-2011]. MOF provides a language for defining metamodels, UML and OCL are well known standards for software modeling and to describe expressions on models, XMI allows MOF-based models to be serialized facilitating the

exchange of models via XML documents, and QVT provides standard languages for transformations in the context of MDA.

MDA defines three different types of conceptual models: Computation Independent Model (CIM) for requirements modeling, Platform Independent Model (PIM) for describing the solution to the problem (design) and Platform Specific Model (PSM) where specific platform details for the implementation are defined, providing a view on software and systems development, based on a series of refinements between these models, by means of automatic transformations and by using the mentioned standards. The Eclipse Modeling Framework (EMF) supports MDA with ECORE for metamodeling based on EMOF (from MOF). In Figure 3.35 the relationships between MDE, MDD and MDA are shown graphically.



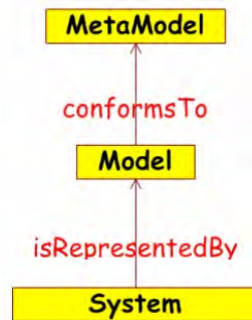
**Figure 3.35.:** MDE, MDD and MDA relationships

Model-driven approaches allow benefits such as reuse at the domain level, increase quality as models are successively improved, reduce costs by using an automated process, and increase software the longevity of solutions, so models become assets instead of expenses [Mellor et al., 2003]. They also allow domain-experts to specify their knowledge and enable technology people to define how this is implemented (using model transformations), providing different implementations for the same model, to run it on different platforms (.NET, Java, CORBA), with integration of existing systems (COTS, legacy systems) [Vallecillo, 2000-2011].

### 3.3.2. Models, Metamodels and Transformations

Models and metamodels are key elements in model driven approaches as mentioned previously. In [OMG, 2003] a model of a system is “a description or specification of a system and its environment for some certain purpose”. In [Mellor et al., 2003] a model “is a coherent set of formal elements describing something (for example, a system, bank, phone, or train) built for some purpose that is amenable to a particular form of analysis .. we express a model in a language that exists at some (language) abstraction level”, and “a description of (part of) a system written in a well-defined language” [Kleppe et al., 2003], where a well defined language provides accurate syntax and semantics and can be interpreted and used by a computer. In [Bézivin, 2005] a complete discussion on models and metamodels is provided. The term “model” is then used in this thesis as a description or specification of (part) of reality (system), written in a well-defined language, that provides a simplified view of it for a certain purpose.

To be able to create models that are able to express the elements being modeled unambiguously, a modeling language’s syntax and semantics must be defined, by building a model of the modeling language, that is to define the so-called metamodel [Mellor et al., 2003], which “describes concepts that can be used for modeling the model” [Stahl and Volter, 2006]. A model conforms to its metamodel i.e. it is written in the (graphical) language defined by its metamodel [Bézivin, 2005]. When models become the center of development, the need to specify them correctly and without ambiguity increases, since syntactic incompleteness makes it impossible to process the model [Karakostas and Zorgios, 2008], which means that metamodels become the key. The relationships between system (reality)-model-metamodel is shown in Figure 3.36.

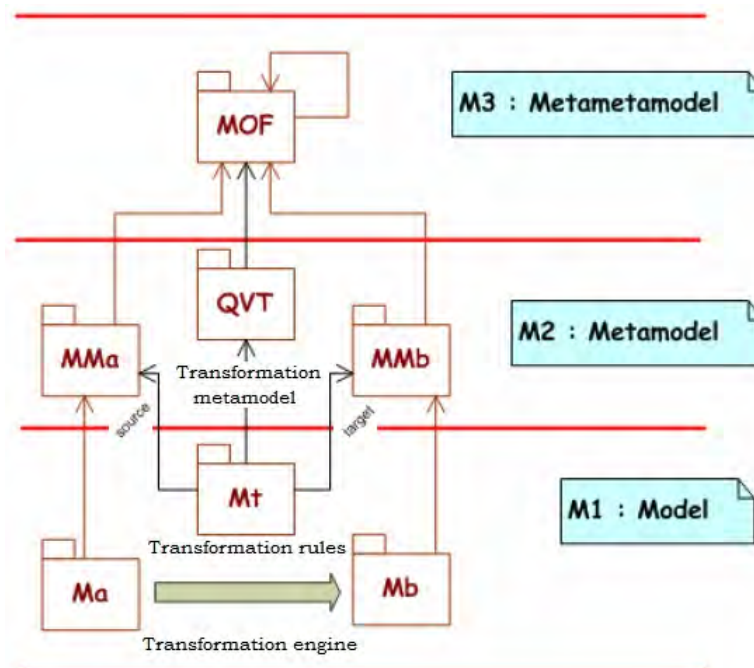


**Figure 3.36.:** Relationships between system-model-metamodel from [Bézivin, 2004]

MOF [OMG, 2011b] from OMG provide a four-layered architecture for organizing models and metamodels manipulation, as well as providing support to the MDA vision, and is described in subsection 3.2.3.

Another key element in a model-driven approach is the definition and execution of transformations to obtain a model from another one, and to generate code from a model for a specific platform. To be able to do this transformation languages are needed which make it possible to define correspondences between the source and the target elements and to support the automatic generation of artifacts. As defined in [OMG, 2003] a model transformation “is the process of converting one model to another model of the same system”, in [Kleppe et al., 2003] it is an “automatic generation of the target model from a source model, which conforms to the transformation definition”, model-to-model (M2M) being the transformation approach.

In this thesis model transformation is as defined in [Bézivin, 2005], where transformation rules correspond to the transformation model which is located at the same level as the source and target models used in the transformation; the transformation model conforms to its defined metamodel (in the case of OMG to QVT), which is at the same level as the metamodels to which the source and target models conform, and a transformation engine supports the execution of the process to generate the target model from the source model, as defined in the model transformation. This view is similar and extends the one provided in OMG [2003] (section 5.4). In Figure 3.37 these concepts are presented in diagram form.



**Figure 3.37.:** Model transformation adapted from [Bézivin, 2005]

Another kind of transformation is the model-to-platform transformation, with which artifacts that are based on the platform are generated from a model, also often called model-to-code or model-to-text (M2T) transformation, and a target metamodel is not needed since it usually deals with simple text replacements [Stahl and Volter, 2006].

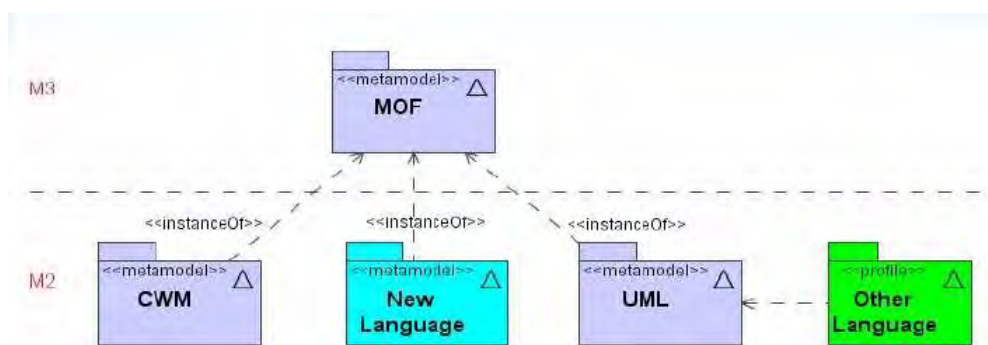
Transformations are classified as horizontal or vertical France and Bieman [2001], where in horizontal transformations the source and target models are at the same level of abstraction (i.e. in MDA from PIM to PIM), for example to change the language in which a model is defined, and in vertical transformations the navigation is from one level of abstraction to another level of abstraction (i.e. in MDA from PIM to PSM), for example to successively refine the level of detail provided in a model.

Several languages exist to support the definition of transformations and tools that allow also their execution, such as the OMG QVT for M2M transformations and the MOF Model To Text Transformation Language (MOFM2T) for M2T transformations or the ATL (ATLAS Transformation Language) [Jouault and Kurtev, 2005] developed by INRIA and integrated in Eclipse, the VIATRA2 (VIsual Automated model TRAnsformations) <sup>10</sup> a sub-project of Eclipse, or GReAT (Graph Rewriting and Transformation) <sup>11</sup> which is part of the GME, to name some of them.

As from the beginning MINERVA was defined to be a standardized framework, we have integrated an MDA approach, using the QVT language to define transformations from BPMN2 models to SoaML service models by means of correspondences between both metamodels as described in chapter 8, and the implementation selected to support the definition and execution of the transformations is the Eclipse MediniQVT plug-in as described in chapter 9.

### 3.3.3. UML Profiles vs. DSL

Modeling based on UML can be too general for certain purposes and domains, that need more “tailored” models and generation of specific artifacts. Based on MOF architecture and definitions (see subsection 3.2.3), MOF and UML can be the basis for defining new languages, to support particular domains: DSLs, which comprise the definition of new metamodels (at the same level as the UML metamodel) to define the corresponding modeling language as described previously, with its own syntax and semantics; and UML profiles which define UML specializations for specific domains, by extending or customizing the UML metamodel (also at the same level as the UML metamodel) [Vallecillo, 2000-2011]. In Figure 3.38 the two options are shown as related to MOF architecture and to MOF and UML.



**Figure 3.38.:** MOF and UML use for defining new languages from [Vallecillo, 2000-2011]

DSLs make key aspects of a domain expressible and modelable by defining a new language to model domain concepts and rules, restricted (in general) to a small set of elements needed for such a domain. Metamodels describing DSLs define the relationships among the defined concepts in the domain and specify the semantics and constraints associated to them [Schmidt, 2006]. Domain

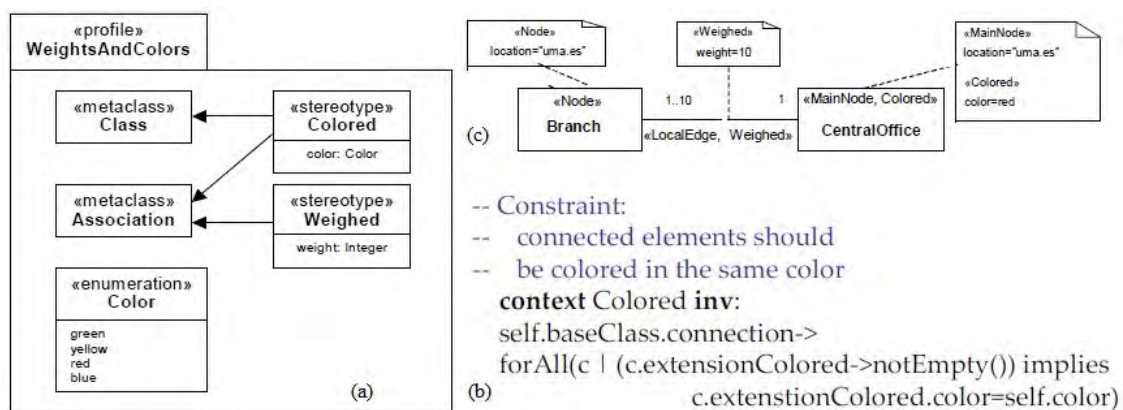
<sup>10</sup><http://eclipse.org/gmt/VIATRA2/>

<sup>11</sup><http://www.isis.vanderbilt.edu/tools/GReAT>

elements are captured as first-class objects and the DSL is itself a model defining all the possible models that developers can build with it [Balasubramanian et al., 2006]. An example of a well-known DSL is the Structured Query Language (SQL) for modeling databases. In the context of this thesis the existing SoaML UML profile is used so we will concentrate below on clarifying UML profile concepts.

UML profiles specialize UML for specific domains by adding stereotypes which “define how an existing metaclass may be extended” [Vallecillo, 2000-2011], adding domain specific terminology. Several uses for UML profiles are mentioned in [Vallecillo, 2000-2011], such as giving a terminology adapted to a particular platform or domain (i.e. EJB terminology: home interfaces, EJBs), giving a different notation for already existing symbols (e.g., use a picture of a computer instead of the ordinary node symbol), or add information to be used when transforming a model to another model or code (i.e. defining mapping rules between a model and Java code).

As defined by OMG <sup>12</sup> a UML profile identifies a subset of the UML metamodel, specifies common model elements expressed in terms of the profile, specifies “well-formedness rules” apart from the ones in the UML metamodel corresponding subset, meaning a set of constraints written in OCL that contributes to the definition of a metamodel element, specifies “standard elements” apart from the ones in the UML metamodel subset, meaning describing a standard instance of a UML stereotype, tagged value or constraint, specifies semantics expressed in natural language, beyond those specified by the identified subset of the UML metamodel. In Figure 3.39 an example of the definition of a UML profile is presented.



**Figure 3.39.:** UML profile definition (a) (b) and use (c) example adapted from [Vallecillo, 2000-2011]

Existing UML profiles defined as OMG standards include: UML SoaML profile, Enterprise Application Integration (EAI), Enterprise Distributed Object Computing (EDOC), CORBA, OMG Systems Modeling Language (SysML), UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) and UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms, among others. The definition of the UML SoaML profile was presented in subsection 3.2.3. In Table 3.1 pros and cons of both approaches for the definition of languages, DSLs and UML profiles, are presented.

<sup>12</sup>[http://www.omg.org/technology/documents/profile\\_catalog.htm](http://www.omg.org/technology/documents/profile_catalog.htm)

**Table 3.1.:** Pros and cons of DSLs and UML profiles from [Vallecillo, 2000-2011]

	DSLs	UML profiles
PROS	More flexibility and control	Easy to start with existing tools
	No dependency on the language defined by the vendor	People think they know UML
	No OMG/standardization dependency	They have already a “standard” UML model to annotate
CONS	Close to the domain	No other modelling tool to use
	New tools are needed	Tools do not know how to deal with a stereotyped element Moving to another tool is difficult
	New capabilities are needed	Profiles are limited in extending and they are only additive, you cannot hide something
	Learning curve for defining them, not for using (or at least what people think)	Defining good UML profiles take more time
	Necessity to maintain homegrown technology	Imprecise UML semantics

### 3.3.4. Main standards for model-driven

In this section three main standards related to the MDA model-driven approach as integrated and used in this thesis are presented: the Model Driven Architecture (MDA) defining the base approach, the Meta-Object Facility (MOF) defining the basis for metamodeling for supporting MDA and the Query/Views/Transformations (QVT) language for the definition of model-to-model transformations.

#### 3.3.4.1. Model Driven Architecture (MDA)

MDA is built on the “well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform” [OMG, 2003], providing an approach for and enabling tools support for: specifying a system independently of the platform supporting it, specifying platforms, choosing a platform for a system and transforming the system specification into one for a specific platform [OMG, 2003]. The principle of separation of concerns is applied by defining three types of conceptual models, described in the following, as defined by MDA [OMG, 2003], in order of level of abstraction from the highest level to the lowest level:

- 1. Computation Independent Model (CIM):**

focuses on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined.

- 2. Platform Independent Model (PIM):**

focuses on the operation of a system while hiding the details necessary for a particular platform, showing that part of the complete specification that does not change from one platform to another. A PIM may use a general purpose modeling language, or a language specific to the area in which the system will be used.

- 3. Platform Specific Model (PSM):**

combines the PIM with an additional focus on the detail of the use of a specific platform by a system. A PIM is transformed into one or more PSMs, where for each specific technology platform, a separated PSM is generated.

Model transformation is the process of converting one model to another model of the same system [OMG, 2003], and a mapping is a set of rules and techniques used to modify one model in order to get another model [Karakostas and Zorgios, 2008]. Transformations and mappings are defined and exemplified in [OMG, 2003] from PIM to PSM and these were the ones mostly applied in the last decade, although in recent years several proposals have also included transformations from CIM to PIM.

As the CIM can be viewed as equivalent to domain modeling, PIM can be viewed as equivalent to design and PSM can be viewed as equivalent to implementation, the difference is that when transforming from PIM to PSM both models are part of the solution space, but when transforming from CIM to PIM the navigation is from the problem space to the solution space, which increases its complexity. In this thesis we propose transformations as the metamodel type for navigating from CIM to PIM as presented in chapter 8.

#### 3.3.4.2. Meta-Object Facility (MOF)

The MOF [OMG, 2011b] standard was first released at the end of the nineties, resulting from the fact that UML was only one of the metamodels in the software development landscape, so the presence of a variety of different incompatible metamodels being defined and evolving independently defined the need for an integration framework for all meta-models in the software development industry. The proposed solution by OMG was the definition of MOF, a language for defining metamodels, i.e. a meta-metamodel [Bezivin and Gerbe, 2001].

MOF defines a four-layered Architecture which defines the abstraction levels for models and meta-models:

- **M3 - Meta-metamodel level**

This is the highest level of abstraction in which MOF itself resides and is defined by itself. MOF is a language for defining metamodels.

- **M2 - Metamodel level**

At the metamodel level metamodels are specified by means of MOF, for example UML. As presented previously defining a metamodel is defining a (modeling) language to write models.

- **M1 - Model level**

At the Model level models for different domains are written by means of the metamodels (languages) defined in the M2 level, for example a domain model in UML.

- **M0 - Instances level**

At the Instances level the occurrence of real elements resides, corresponding to a given model from the M1 level. For example customers in a bank.

As defined in [OMG, 2011b] “The MOF 2 Model is used to model itself as well as other models and other metamodels (such as UML 2 and CWM 2 etc.). A metamodel is also used to model arbitrary metadata (software configuration or requirements metadata, for example). Metamodels provide a platform independent mechanism to specify the following: the shared structure, syntax, and semantics of technology and tool frameworks as metamodels; a shared programming model for any resultant metadata (using Java, IDL, etc.); a shared exchange format (using XML)”. MOF is defined in two parts: Essential MOF (EMOF) which acts as a kernel metamodeling capability and Complete MOF (CMOF) which ultimately is MOF. In Figure 3.40 the four modeling layers of MOF are presented in diagram form.



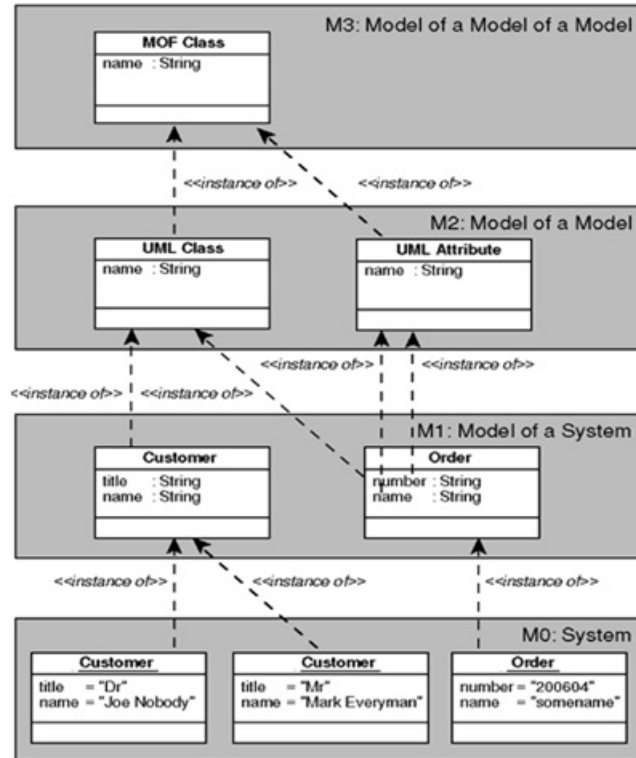


Figure 3.40.: Four modeling layers of MOF from [Kleppe et al., 2003]

The use of the four-layered architecture from MOF in MINERVA is presented in chapter 4 indicating for each level the models and metamodels we have integrated.

### 3.3.4.3. Query/Views/Transformations Language (QVT)

QVT [OMG, 2008c] provides the definition of languages for specifying transformations between MOF models in a standard way in the context of the MDA approach. It has a hybrid declarative/imperative nature, defining three metamodels one for each language organized in one package each: QVTCore, QVTRelation and QVTOperational, the first two being declarative and the last one imperative.

The Relations metamodel and language provides support for object pattern matching and object template creation, and traces between model elements in the transformation are created implicitly. The Core metamodel and language is defined using minimal extensions to EMOF and OCL, and trace classes are explicitly defined as MOF models. The Operational language provides imperative implementations of transformations from Relations or Core, which populate the same trace models as Relations. Another mechanism is provided for invoking imperative implementations the Black-box MOF Operation implementations, making it possible to “plug-in” an implementation of a MOF Operation in any other language that can be bound to MOF [OMG, 2008c]. In Figure 3.41 the relationships between QVT metamodels is shown.

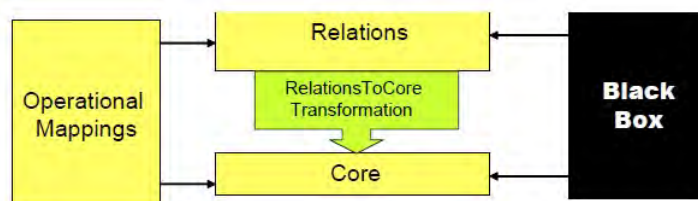


Figure 3.41.: Relationships between QVT metamodels from [OMG, 2008c]

In the following the Relations language which is the one we have used to define QVT transformations in MINERVA as presented in chapter 8, is described briefly.

The Relations language allows transformations between models to be specified as a set of relations that must hold for the transformation to be successful. Each model conforms to a model type (the metamodel) specifying the elements that any conforming model can have and which are contained within the packages referenced from the transformation. Relations in a transformation are defined by two or more domains and a pair of when and where predicates, specifying a relationship that must hold between elements of the involved models. A domain is a distinguished typed variable that can be matched in a model of a given model type. The when clause specifies the conditions under which the relationship needs to hold, that is the preconditions, and the where clause specifies the condition that all model elements participating in the relationship must satisfy; both allow any OCL statement [OMG, 2008c]

Whether or not a relationship may be enforced is determined by the target domain, which may be marked as checkonly or enforced. When it is checkonly then it is only checked to see if there exists a valid match in the model satisfying the relationship, when enforced, if checking fails then the target model is modified to make the relation hold by creating, deleting or modifying only the target model, enforcing the relationship. There are two types of relations defined: top-level relations that must all hold for the transformation, and non-top-level relations which must hold only when invoked directly or transitively from the where clause of another relation.

Pattern matching associated with domains is known as object template expressions which are used to match patterns in candidate models, in other words, to find coincidences of elements in the models that are to be transformed or compared. A template expression match results in a binding of model elements from the candidate model to variables declared by the domain. If a binding already exists for a variable to model elements for example from the evaluation of a when clause, only bindings for free variables of the domain are matched, i.e. existing bindings are preserved. Pattern matching allows the creation of objects in the target model when a valid match of the target domain pattern does not exist, and when objects already exist they are updated, for which the concept of “Key” is used to define a set of properties of a class that uniquely identify an object instance of the class in a model, so avoiding the creation of duplicated objects. In Figure 3.42 the QVTBase package which contains common structures defined for transformations and rules, is presented, and in Figure 3.43 the QVT Relations package is shown.

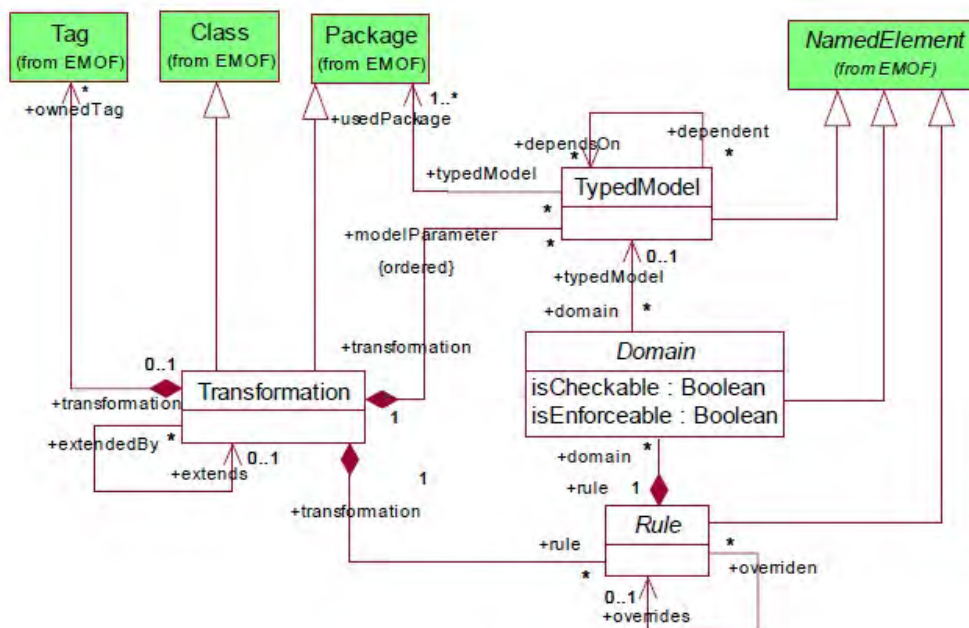


Figure 3.42.: QVTBase package - transformations and rules from [OMG, 2008c]

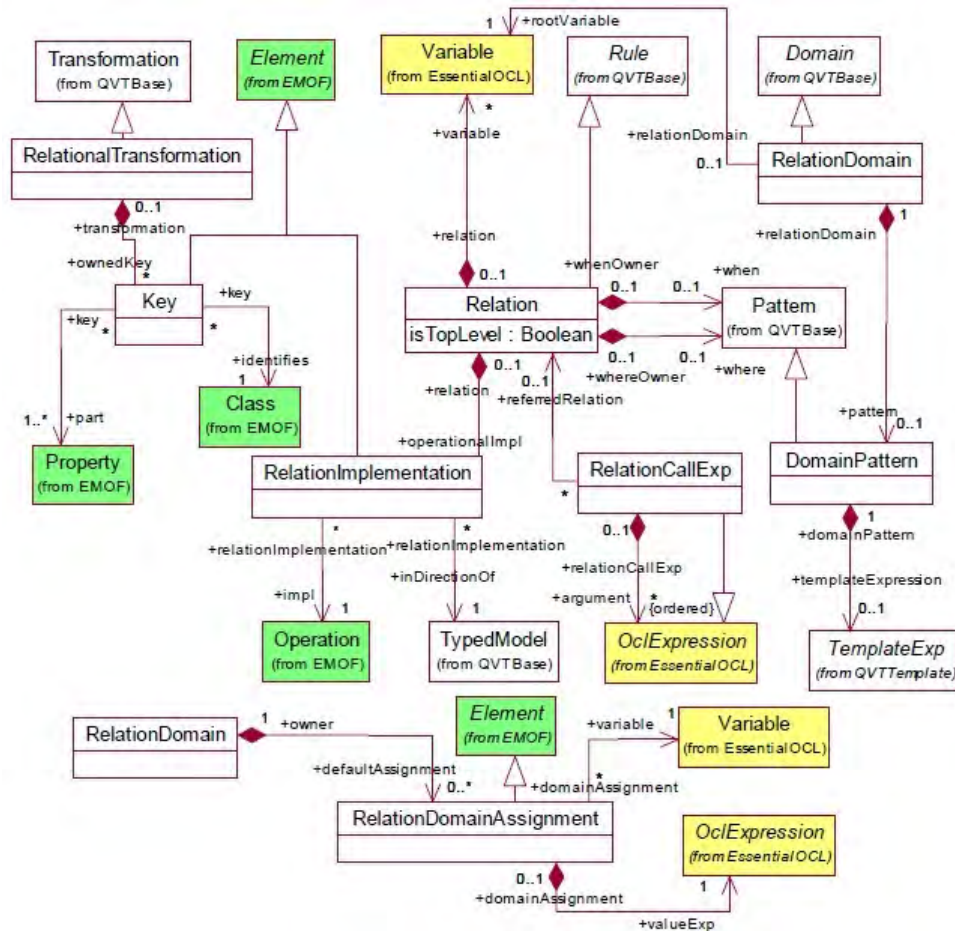


Figure 3.43.: QVT Relations package from [OMG, 2008c]

The Operational Mapping language makes it possible to define transformations in an imperative approach called operational transformations, or complementing relational transformations with imperative operations implementing the relations known as a hybrid approach. The complete definition of the three languages in the QVT standard can be checked in [OMG, 2008c].

### 3.4. Continuous Process Improvement (CPI)

Several improvement initiatives have surfaced over the last decades such as JIT (Just In Time), TQM (Total Quality Management), Lean manufacturing, BPR (Business Process Reengineering), and Six Sigma [Laguna and Marklund, 2005]. We will not discuss each of these initiatives but rather take a high level view on defining CPI and discuss different types of improvement (Evolutionary vs. Revolutionary), the focus on execution measurement as the core of any BP improvement program. Finally a key standard for BP improvement is presented, BPMM [OMG, 2008b].

#### 3.4.1. BP improvement concepts and types

Organizations in different domains such as software, manufacturing, marketing, banking, and finance, share similar problems: overworked staff due to poor estimating and planning; excessive rework; lack of consistent and stable processes often with multiple ways to do similar things; lack of basis for measurement and management; lack of foundation for organization-wide approaches and solutions; disappointing results from automation; mixed results when applying approaches such as Six Sigma or BPR and improvements that are too localized and sub-optimal from a global business perspective [OMG, 2008b].

One key statement of process management is that quality of products and services is largely determined by the quality of the processes used to develop, deliver and support them, where an effective process is capable of bringing people, tools and methods together into an integrated whole which produces the expected outcomes [OMG, 2008b]. “...successful process management is at the core of all process-oriented improvement programs that have surfaced over the last couple of decades” [Laguna and Marklund, 2005].

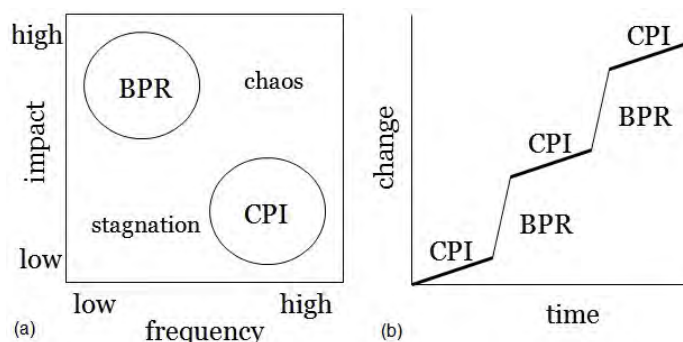
Continuous process improvement refers to a status in which the organization is continuously analyzing the way it carries out its business, finding improvement opportunities for performing its BPs [OMG, 2008b]. An improvement effort has to support the identification of process deficiencies and provide guidance for introducing improvements in a systematic way, for which measures of the BP, activities, performance, resources, cost and results have to be defined, implemented, collected and analyzed on a regular basis [OMG, 2008b].

An improvement program has to provide specific artifacts and a systematic way to support and guide the improvement effort in the organization. It is not enough to provide measures and means to analyze them, including tool support, but it is essential to align measures with business strategy and goals for the entire organization with the ones specific for each BP. This will allow the information collected from their execution to be interpreted correctly. The primary focus should be first on understanding and defining BPs business goals (i.e. performance, results, costs, etc.) and on measuring their business results against these goals [OMG, 2008b].

Two main approaches exist for the introduction of changes in BPs in organizations: evolutionary and revolutionary change. The evolutionary change model advocates that change should come from within the organization itself, managed by the current leadership and carried out by the current employees, assuming that real change is best achieved through incremental improvements over time [Laguna and Marklund, 2005]. Both CPI and TQM share this philosophy for introducing changes.

On the other hand, the revolutionary change model unfolds quickly, altering the structure of the organization, business practices and culture, and needs to be top driven typically by the CEO and externally imposed on the organization, with external resources (consultants) and an outside viewpoint [Laguna and Marklund, 2005]. BPR which promotes this philosophy, has as its essence the aim to achieve drastic improvements by completely redesigning core BPs, in other words, rethinking the way business is conducted, and introducing fast revolutionary implementation [Laguna and Marklund, 2005].

As mentioned in section 3.1 BPR was promoted by the publication of articles at the beginning of the nineties by [Davenport and Short, 1990] and [Hammer, 1990] and latter books, at a time when business were looking for answers on how to compete effectively in the changing marketplace, so it was embraced. However, between fifty and seventy percent of reengineering projects have failed to achieve the results defined in their objectives [Laguna and Marklund, 2005]. While some attribute the failures to senior management, they can also be attributed to a misunderstanding of the underlying philosophy. In Figure 3.44 main differences between CPI and BPR are shown: (a) impact of changes over frequency of change and (b) changes over time.



**Figure 3.44.:** CPI and BPR differences (a) impact of changes, (b) changes over time from [van der Aalst, 2002]

As can be seen in Figure 3.44 (a), while the frequency for integrating changes is high in CPI, meaning continuous introduction of small changes, the impact of each integrated changes is low, which is the opposite to what occurs with BPR: while the frequency for integrating changes is low, meaning disrupted integration of radical changes, the impact of changes is high. When analyzing (b) the achievement of changes over time is also different, while CPI moves with small changes with a long duration for achievement, BPR moves with high changes over a short duration. However, as both are process centric they can be supported by a BPMS (WFMS) [van der Aalst, 2002].

Improvement approaches such as TQM are usually based on an improvement cycle such as the Deming cycle: Plan-Do-Check-Act (PDCA) or adaptations such as the Define-Measure-Analyze-Improve-Control (DMAIC) from Six Sigma, to name some of them. In MINERVA a CPI philosophy is integrated in the definition of BPCIP as presented in chapter 5, as we believe that such an evolutionary path through improvement is more suitable for generating a continuous learning and improvement status in the organization, to be sustained over time.

### 3.4.2. BP execution measurement

Measurement of BPs execution provides the basis for analyzing the real behavior of BPs in the organization, helping detect deviations from the planned behavior, which will lead to finding improvement opportunities for the BPs. Execution measurement then becomes the enabler towards understanding and controlling the real occurrences of BPs in the organization, to establish a continuous BP improvement culture [OMG, 2008b].

There is much literature on the definition and calculation of BP execution measures, focusing mainly on the analysis of performance (time, capacity) and cost, using analytic techniques and simulation, for example to evaluate possible execution scenarios for the BP model, as in [Laguna and Marklund, 2005, Reijers, 2003, Netjes, 2010, zur Muehlen, 2004]. Analytic techniques to analyze the behavior of BPs are based on mathematical theories and formalisms, such as queuing theory or Markov chains. When used for measuring BP execution, where the real data of execution is available, the formulae presented here are adapted, as described in chapter 6.

To calculate times of BPs execution, several elements of the BP control flow must be taken into account, such as activity execution states and times, and specific constructions for the divergence of the control flow into different paths, as set out below. After these, commonly used time and cost measures are presented, and finally execution event logs are described.

#### 3.4.2.1. Activity instances lifecycle and execution times

In the first place, the execution times for the activities includes processing (or service) times and waiting (and queuing) times (when applying analytic techniques these times are usually given as averages including waiting times). These times are related to the lifecycle of activities in which there is a definition of the states in which an activity may be when a BP is executing. An example of an activity lifecycle is presented in Figure 3.45 (a) and in (b) related execution times.

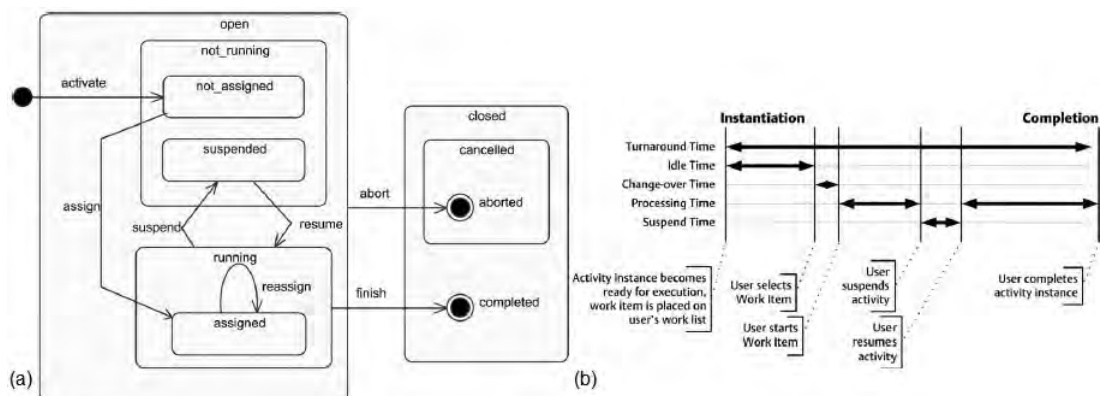
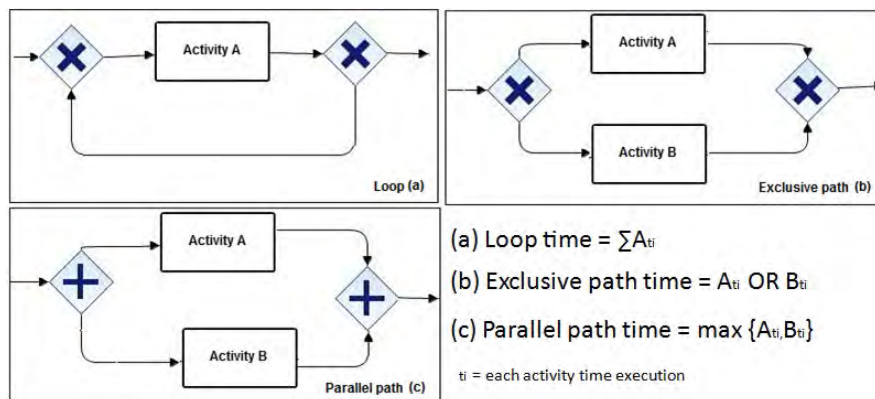


Figure 3.45.: (a) Activity lifecycle and (b) execution times from [zur Muehlen, 2004]

In Figure 3.45 (a) and (b) it can be seen that after an activity is enabled (activate) it becomes ready for execution, but until it is actually started idle time is counting (waiting time). After it is started processing (working) time begins to count until the activity is either completed or suspended, in the first case the activity is finished, and in the second one suspend (waiting) time is counting again. Once the activity is resumed processing (working) time begins to count again, until the activity is completed. BP cases also have an associated lifecycle that is related to the one for activities reflecting the state of the activities in the BP case, for example if an activity is “running” the BP would be for example “in progress” or “started”.

### 3.4.2.2. Control flow view on execution times

The control flow view in BP models defines the order in which the activities can be executed and bifurcations that may occur in the flow, several workflow patterns [van der Aalst et al., 2003a] have been defined to express these modeling options. For the calculation of time execution measures three key patterns have to be taken into account as there are specific formulae to be used when calculating them. Figure 3.46 presents these three patterns and the corresponding formulas to calculate their associated times, based on [Laguna and Marklund, 2005].



**Figure 3.46.:** Use of Gateways and formulas to calculate times [Laguna and Marklund, 2005]

The patterns presented are: (a) loop where an activity or several activities may be re-executed when some business rule defines a control point after the activities in the loop are executed, in order to evaluate if the flow can continue or has to be sent backwards; (b) exclusive path where only one branch of the decision point can be followed so the activities in the branch that is not taken will not be executed at all, and (c) parallel path where all the branches defined by the decision point will be executed.

In (a) all execution times for the activity/ies executed within the loop has to be added up to obtain the execution time for the loop (when applying analytic techniques the probability of executing the loop is known), in (b) only the executed path are added up (when applying analytic techniques the probability of selecting each of the path options is known) and (c) where only the maximum of the execution times for all paths is taken into account. For other patterns the calculation is similar, for example to calculate the times for the Inclusive path the formula is the same as for the parallel path but taking into account only the branches that have been executed.

For the calculation of the process capacity the probabilities of traversing each branch have to be known in two cases: when the pattern is the exclusive path (b) as the BP cases flow differently between the options; and for loops (a) the probability of rejection that is the flow being sent back to execute the activities in the loop again; the parallel path construction has a probability of 1 for each of its branches as all BP cases flow for all defined branches.

### 3.4.2.3. Time and Cost execution measures

Throughput time (TT) (or Cycle time, CT) is one of the most important measures of performance of a BP, and corresponds to the time that it takes to complete an individual BP case from start to finish [Laguna and Marklund, 2005], or the BP case total amount of time spent from the moment that the case is initiated until the time it is completed [Reijers, 2003]. The average TT is the result of the sum of the individual TT for a set of BP cases divided by the total number of BP cases, which in queuing theory is referred to as the expected time in the system [Laguna and Marklund, 2005]. The objective is to minimize the TT of the BP.

The capacity of the BP refers to the number of BP cases each resource type can handle per unit of time, for example a role Secretary. To calculate the capacity of the BP, in the case of applying analytic techniques) for each activity the processing time, the type of resource required, the number of available resources of each type and the number of BP cases processed through each activity must be known. In the case of execution measurement the processing time of the activity can be replaced for the total time including waiting times.

Capacity is associated with resources or resource types, and determines the bottleneck of the process, which is determined to the smallest resource type capacity (pool capacity). Its calculation is also affected by the control flow constructions loop, parallel path and exclusive path. For each resource type its capacity is calculated as (taken from [Laguna and Marklund, 2005]):

*Resource type capacity = Unit capacity (1/Unit load) x number of available resources of this type*

where *Unit load =  $\Sigma$  (processing time activity x number of BP cases activity) for each activity that the resource is required*

The objective is to maximize the capacity of the process, for which the obvious action to be taken will be to add resources to the bottleneck, although this sometimes is not possible and other solutions must be provided.

Calculation of cost can be associated with different perspectives such as fixed and variable costs, where the first refers to overhead cost which are not affected in general by the processing of the BP, for example use of infrastructure; variable costs are correlated with the quantity of a variable for example level of sales. Operational cost is the one that can be directly related to the outputs of a BP, where an important part corresponds to labor cost, that is the cost related to human resources working in the BP [Reijers, 2003].

These execution measures provides the basis for the set of execution measurement we have integrated to support the measurement activities defined in MINERVA, which will be described in chapter 6.

### 3.4.2.4. BP execution event logs

To be able to analyze BPs execution and calculate execution measures data from the real execution of BPs must be registered as the BP cases executes. The execution of BPs in process engines is usually registered based on activity times as presented previously, such as start and complete, the resource (human) executing the activities, and other information such as the ID of the activity instance, the ID of the BP case, and the ID of the associated BP model to which the BP case corresponds. Each process engine allows execution data to be extracted in different ways and formats, which can then be transformed into the format used by other tools such as the ProM<sup>13</sup> framework to be able to analyze BPs execution. The general structure of a typical execution event log is presented in Table 3.2.

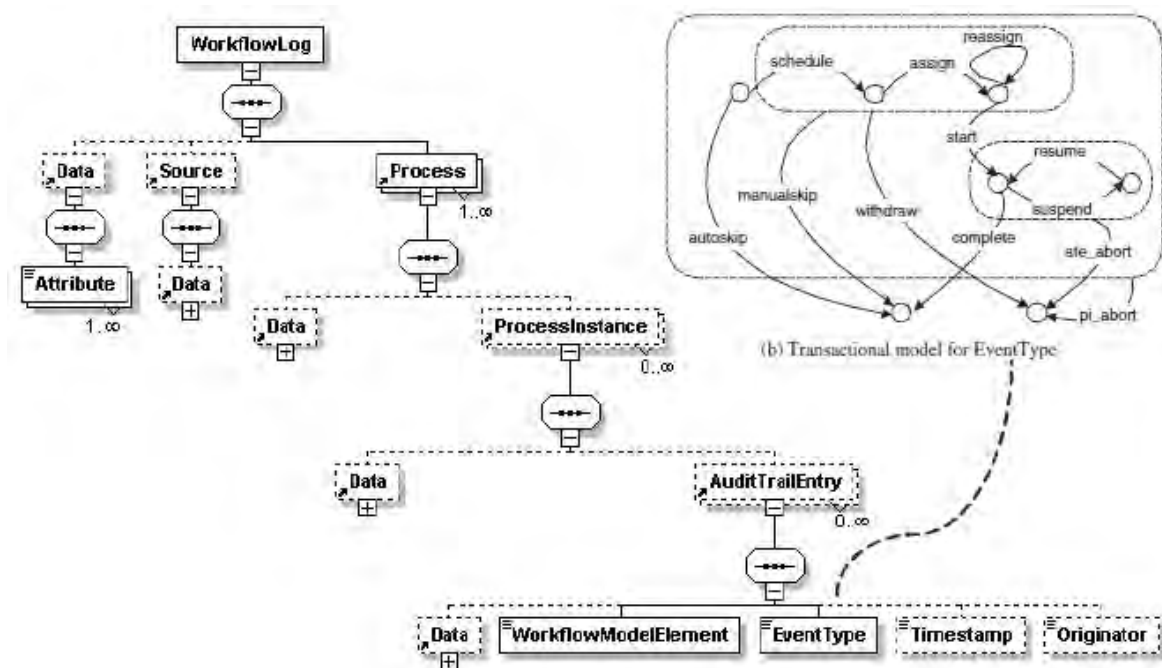
---

<sup>13</sup><http://www.processmining.org/prom/start>

**Table 3.2.:** General structure of a typical execution event log based on [van der Aalst et al., 2007]

BP case id	Activity id	Originator	Event	Timestamp
case 1	activity A	John	start	11-11-2011:11.11
case 1	activity A	John	complete	11-11-2011:11.15
case 2	activity A	John	start	11-11-2011:11.22
case 3	activity A	Sue	start	11-11-2011:11.25
case 3	activity A	Sue	complete	11-11-2011:11.29
case 2	activity A	John	complete	11-11-2011:11.30
case 3	activity B	Carol	start	11-11-2011:11.40
case 1	activity B	Mike	start	12-11-2011:09.22
case 1	activity B	Mike	complete	12-11-2011:09.45
case 3	activity B	Carol	complete	12-11-2011:10.00

For the ProM framework a specific structure for execution event logs has been defined called MXML<sup>14</sup> (Mining XML) which has been superseded by XES<sup>15</sup> (eXtensible Event Stream) by the end of 2010, although MXML is still in use. MXML specifies the process log format, being the root element a WorkflowLog element, which contains: optional Data and Source elements, and a number of Process elements. A Data element is used for storing textual data and contains a list of Attribute elements, a Source element can store data about the information system originating the log, and a Process element refers to a specific process in it. A ProcessInstance is a BP case, an AuditTrailEntry can be an activity (WorkflowModelElement), and event type (Eventtype), a timestamp (Timestamp) and the person that executed the activity (Originator) [van der Aalst et al., 2007]. In Figure 3.47 the Process log XML format (a) and the transactional model (b) for event types is shown.

**Figure 3.47.:** Process log XML format (a) and transactional model (b) for event types from [van Dongen et al., 2005]

The transactional model represents the defined activity lifecycle as presented previously, in which the allowed states are modeled in a similar way. An example of an MXML execution event log is shown in Figure 3.48.

<sup>14</sup><http://www.processmining.org/logs/mxml>

<sup>15</sup><http://www.processmining.org/logs/xes>



```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- MXML version 1.0 -->
<!-- Created by Fluxicon Nitro (http://fluxicon.com/nitro/ -->
<!-- (c) 2010 Fluxicon Process Laboratories / http://fluxicon.com/ -->
<WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd">
  <Source program="Fluxicon Nitro"/>
  <Process id="running-example.mxml" description="Converted to MXML by Fluxicon Nitro">
    <ProcessInstance id="3">
      <AuditTrailEntry>
        <Data>
          <Attribute name="Activity">register request</Attribute>
          <Attribute name="Resource">Pete</Attribute>
          <Attribute name="Costs">50</Attribute>
        </Data>
        <WorkflowModelElement>register request</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>2010-12-30T14:32:00.000+01:00</Timestamp>
        <Originator>Pete</Originator>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <Attribute name="Activity">examine casually</Attribute>
          <Attribute name="Resource">Mike</Attribute>
          <Attribute name="Costs">400</Attribute>
        </Data>
        <WorkflowModelElement>examine casually</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>2010-12-30T15:06:00.000+01:00</Timestamp>
        <Originator>Mike</Originator>
      </AuditTrailEntry>
    </ProcessInstance>
    <ProcessInstance id="2">
      <AuditTrailEntry>
        <Data>
          <Attribute name="Activity">register request</Attribute>
          <Attribute name="Resource">Mike</Attribute>
          <Attribute name="Costs">50</Attribute>
        </Data>
        <WorkflowModelElement>register request</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>2010-12-30T11:32:00.000+01:00</Timestamp>
        <Originator>Mike</Originator>
      </AuditTrailEntry>
    </ProcessInstance>
  </Process>
</WorkflowLog>

```

**Figure 3.48.:** An example of an MXML event log from examples of [van der Aalst, 2011]

As can be seen in Figure 3.48 several assumptions are made about event logs [van der Aalst, 2011]: a BP consists of BP cases where all events from each BP case are listed into the corresponding BP case and ordered within it, by means of time occurrence. As mentioned before, events can have attributes such as activity, resource and cost. Data corresponding to the workflow model element, event type, associated timestamp and originator are also present in the event log.

### 3.4.3. Main standards for CPI

In this section a key standard related to BPI is presented, the Business Process Maturity Model (BPMM) as defined by OMG.

#### 3.4.3.1. Business Process Maturity Model (BPMM)

BPMM [OMG, 2008b] is an OMG standard which aims to provide a path for improvement from an immature to a mature status of organizations in which changes are detected and integrated in a proactive way and the expected results from BPs are mostly achieved. BPMM was conceived by the same authors of the Capability Maturity Model Integration (CMMI) and Capability Maturity Model (CMM) from the Software Engineering Group (SEI) for software, and is defined in the same way but changing the focus on definitions to organizational and management issues of BPs.

As is stated in [OMG, 2008b], when there is a lack of objective data (or measures) for judging the quality of products or services delivered, or for solving product, service, or process problems,

an organization is unable to take action based on quantitative and objective information. In a mature organization, historical documented data is available and there is an objective quantitative basis for analyzing those problems, so estimates and plans are based on this data. Amongst other things, this helps achieve the expected results for cost, schedule, performance and quality. When changes are needed to address problems, the different options are understood, as are the overall effects and consequences of taking each one. A systematic improvement effort helps organizations in the path from the immature to the mature status [Sánchez González et al., 2009, OMG, 2008b].

BPMM defines five levels of maturity which assess several characteristics of BPs and defines several practices in Process Areas for each level, which, when applied, make it possible to navigate from a lower maturity level to upper ones. The performance of a process describes the real results that are achieved by means of the process; the process capacity describes the rank of expected results that can be achieved by means of the process (output prediction), where Process Areas and maturity levels are indicators of the process capacity; and process maturity describes the grade in which processes are explicitly defined, managed, measured, controlled and are effective [OMG, 2008b].

As in CMMI and CMM the levels are numbered from one to five, and in BPMM are named as: 1 - Initial, 2 - Managed, 3 - Standardized, 4 - Predictable and 5 - Innovating. In Figure 3.49 the graphical representation and meaning is provided.



**Figure 3.49.:** BPMM maturity levels from [OMG, 2008b]

In level 1 -Initial- process practices and results are inconsistent, there are no defined objectives and results are dependent on persons not on processes, in general the result of processes/projects are unsuccessful and when results are good there is no way to know the reason for the success, as there is no measurement at all.

In level 2 -Managed- every unit of work and/or project has a basic process for planning, management and control of requirements and carries out essential activities to prepare, distribute, operate and support its products and services. Process are defined and repeatable in each unit of work/project, but can be different between different units. In each unit expected results are usually achieved in time and budget fulfilling requirements.

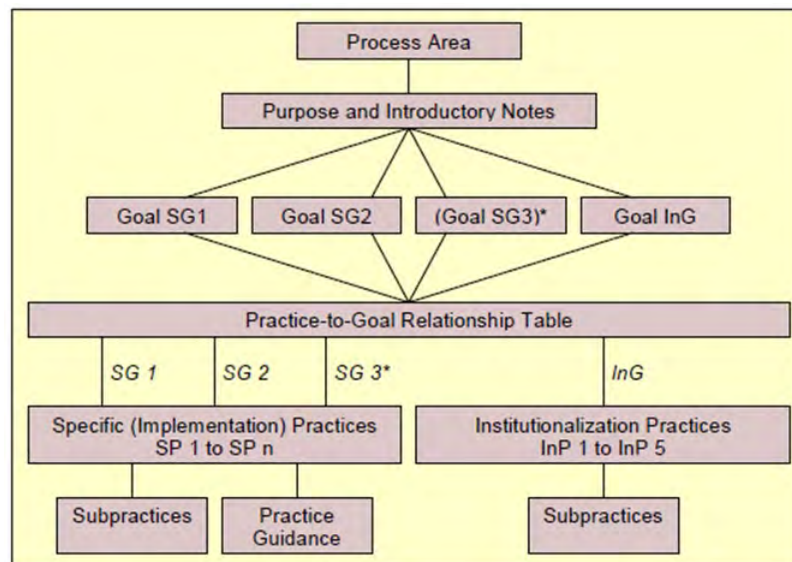
In level 3 -Standardized- the organization has standardized processes to prepare, distribute, operate and support its products and services. This processes are defined at a high level and instantiated for each unit of work/project; the organization as a whole benefits from standardized best practices.

In level 4 -Predictable- the organization defines achievable quantifiable objectives for performance and quality of BPs, based on the standardized organizational processes defined in level 3 with defined infrastructure and assets, to obtain predictable results with controlled variations.

In level 5 -Innovating- the organization has knowledge about its business critical areas and characteristics such as competitiveness and defines quantifiable objectives to introduce improvements

in them. Improvements are identified, evaluated, put into practice in pilot experiences and distributed to the organization to achieve the defined improvement objectives. The main objective is the continuous improvement of processes in the organization and their resulting products and services.

Each maturity level integrates a set of Process Areas, which in turn defines a set of practices for the area that provides a process capacity needed for that level. Each Process Area has a purpose, specific and institutionalization objectives and practices, with sub-practices and realization guides. There are thirty Process Areas defined: nine for level two, ten for level three, five for level four and six for level 5. In the definition of Process Areas is presented.



**Figure 3.50.:** Definition of Process Areas in BPMM from [OMG, 2008b]

A specific guide for measurement activities is provided, as BPMM does not define an specific measurement Process Area such as CMM and in contrast to CMMI, which does so, so the measurement activities are spread in the rest of the Process Areas, making it difficult to gain a global view of the proposed measurement approach. A discussion on this can be seen in [Sánchez González et al., 2009].

Improvements on process performance indicates that the Process Areas practices for the maturity level have been implemented and institutionalized; improvements in the process maturity level indicate that the process capacity has been improved over time (it is possible to predict the outputs of the process better), and improvements in the organization maturity level indicate that methods, procedures and practices have been institutionalized and are independent of people.

BPMM is used in MINERVA to support the evaluation of the maturity level of process in the organization in the context of the improvement efforts carried out by means of BPCIP as described in chapter 5.

### 3.5. Systematic literature review

The systematic literature review [Delgado et al., 2012b] presented in this section corresponds to an update of the systematic literature review [Delgado et al., 2010g] that was carried out with the main objective of providing the basis for the research work for this thesis, assessing the application of service-oriented and model-driven paradigms to business processes. The research path defined for the systematic review is that of the methodological application of service-oriented and model-driven paradigms to business processes. The focus is on disciplined, conceptually based and standardized associated practices, not on a specific tool or interpretation. This allows us to maximize the value

of the understanding and application of these paradigms jointly, to obtain an organization's defined business value.

Following the method proposed by Kitchenham [Kitchenham, 2004, Kitchenham and Charters, 2007], a systematic review consists of three stages: planning the review, development of the review and publication of the results. These stages have several elements, starting from the definition of the research question, key words and research chains, the execution of sources for the defined chains, and the inclusion and exclusion criteria to select relevant and primary studies from which to extract the associated data. The defined objective is to identify the joint application of SOC and MDD paradigms to business processes, which allows us to assess aspects related to their use.

### 3.5.1. Research Question, Search String and Sources

The research question that guides this work includes these specific terms in its initial formulation: SOC, MDD, MDE, BP and BPM, and new terms added in the 2009 update like Service-Oriented Development (SOD). The research question is:

*¿ Which methodological/conceptual applications of Service-Oriented (Computing or Development) and Model-Driven (Development or Engineering) paradigms to business processes and the Business Process Management (BPM) paradigm have been carried out ?*

in technological aspects or specific tools for implementation. The main identified key words for carrying out the searches are: Service-Oriented Computing and Development (SOC, SOD), Model-Driven Development and Engineering (MDD, MDE), Business Process and Business Process Management (BP, BPM). The research question, keywords and search strings were validated by experts. The criteria for selecting the sources included the importance of the source as a repository of scientific articles, possibility of access and web site, resulting in the following selections:

- ACM Digital Library (<http://portal.acm.org>),
- IEEE Digital Library (<http://www.computer.org>),
- SCOPUS (<http://www.scopus.com>),
- Science@Direct in CS area (<http://www.sciencedirect.com>),
- WileyInterScience in CS area (<http://www.interscience.wiley.com>).

The search strings were obtained from the combination of the defined keywords, mainly with the Boolean operands "AND" and "OR". The main search string was constructed by combining the defined key words, obtaining the following general search string, which was adapted for each search engine of the selected sources:

*("service-oriented computing" OR "service-oriented development") AND ("model-driven development" OR "model-driven engineering") AND ("business process" OR "business process management")*

We decided not to include words like "conceptual" and "methodology" since the approach is not generally stated in those terms explicitly, although it clearly appears in a reading of the study.

### 3.5.2. Study Selection and Information Extraction

To select the studies, inclusion and exclusion criteria had to be defined, according to the research question and the search strings defined. The inclusion criteria were first applied to select a reduced set of relevant studies to which to apply the exclusion criteria to select the primary studies, responding to the research question. To be selected, the proposals had to deal with the methodological and/or conceptual application of SOC and MDD paradigms to business processes, presenting a methodology, unified proposal relating the paradigms, transformations and relations between business processes and service models and notations.

Once the articles were filtered by reading the title, abstract and key words of all of the articles obtained, the relevant ones were selected according to the inclusion criteria. The exclusion criteria were then applied, by reading the articles thoroughly to select the primary studies with relevant information about the research issue. Studies were excluded if they focused on technology issues, or composition and/or generation of Web Services (WS) and languages for execution as WS-BPEL or implementation with workflows, integrated other paradigms such as agents or grid computing, or focused on specific issues of each paradigm such as security or variability for business processes, or Quality of Services (QoS) requirements for services, dealt solely with one paradigm or did not include business processes as their main focus.

Table 1 shows the columns corresponding to Found repeated (FR), Relevant Not Repeated (RNR) and Primary (P) studies, for the first search which corresponds to the period 2000 to 2007, the 2009 update which corresponds to the period 2008 to June 2009 and the 2010 update which corresponds to the period July 2009 to December 2010.

**Table 3.3.:** Number of studies obtained from the selected sources

Source	FR	RNR	P	FR	RNR	P	FR	RNR	P
ACM DL	259	3	3	426	3	0	255	8	3
IEEE DL	779	10	8	300	8	4	238	2	2
SCOPUS	264	8	5	327	11	7	294	11	6
Science Direct	503	5	2	499	3	1	255	2	0
Wiley IScience	8	0	0	4	0	0	16	0	0
ST (2000-2007)	1813	26	18	–	–	–	–	–	–
ST (2008-2009)	–	–	–	1556	25	12	–	–	–
ST (2009-2010)	–	–	–	–	–	–	1080	23	11
Total (2000-2010)	–	–	–	–	–	–	4449	74	41

After eliminating the repeated articles and reading the title, abstract and keywords of the resulting articles, 26 were selected as relevant in the first search, and 18 of those were selected as primary studies. In the 2009 update, 25 studies were selected as relevant, of which 12 were primary studies, and in the 2010 update, 23 studies were selected as relevant of which 11 were primary studies, giving a total of 41 primary studies. It worth mentioning that in the 2010 update the article to the first version of the systematic review published [Delgado et al., 2010g] was retrieved in the searches, but it was not included as a primary study, as it is a secondary study. The high number of total articles responds to the fact that many papers refers to SOC and MDD paradigms as well as business processes, although the proposal does not elaborate on their joint application.

Once the primary studies were selected, the relevant information was extracted. To perform this task, a form consisting of many sections was defined, including general data such as title, publication and authors with affiliations and associated country, year of the study, general study description, and an indication of which paradigms it integrates. Several key aspects of paradigm integration were found in this process, for which the studies were classified.

### 3.5.3. Analysis of the Results

This section analyzes and discusses the contents of the selected primary studies in order to extract, organize and present the relevant information. In the first place a summary of the studies is presented, showing the paradigms they integrated, the notations used and the type of case study they provide. Secondly key findings of this systematic literature review are described by presenting the main principles detected for paradigms integration from the selected studies, which we have categorized and discussed. Finally a summary of these principles as found in the selected studies is presented. The data extraction from the primary studies is shown in Appendix A.

### 3.5.3.1. Summary of studies

In this section the selected studies were summarized based on key information we wanted to know, including the paradigms integration that each study deals with, the notations used in each study for BP and services modeling, and the type of case study provided: example or real application.

#### Paradigm integration

Figure 3.51 (a) shows the publications by paradigm integration. It can be seen that of the total number of studies, more than half (51%) correspond to the application of SOC and MDD paradigms to BPs, and the other half is divided between the application of SOC to BPs (29%) and the application of MDD to BPs (20%). These results are consistent with last year's tendency to unify service-orientation and BPs to support organizational needs, and to provide automated support for this integration.

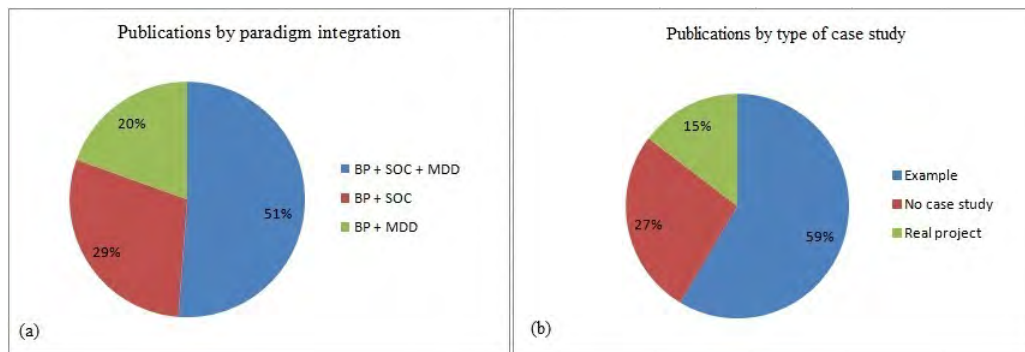


Figure 3.51.: Publications by Paradigm integration (a) and Type of case study (b)

#### Type of Case Study

Figure 3.51 (b) shows the publications by type of case study. Of the total number of studies, 59% correspond to examples prepared to show different aspects of the proposal, generally based on standard BPs used in organizations, and only 15% correspond to real projects in organizations in which the proposal has been used in projects with industry. On the other hand, 27% of the studies do not present a case study. It can be concluded that although some work has been done on the real application of the proposals, more is needed to show the benefits for organizations effectively.

#### Notations Used

Regarding business process and service modeling, one of the most important issues is the notations used to specify them. Many efforts have been made to define notations to support the different views of software development and BPs integration. In Figure 3.52 notations used for BPs and services/software modeling are presented.

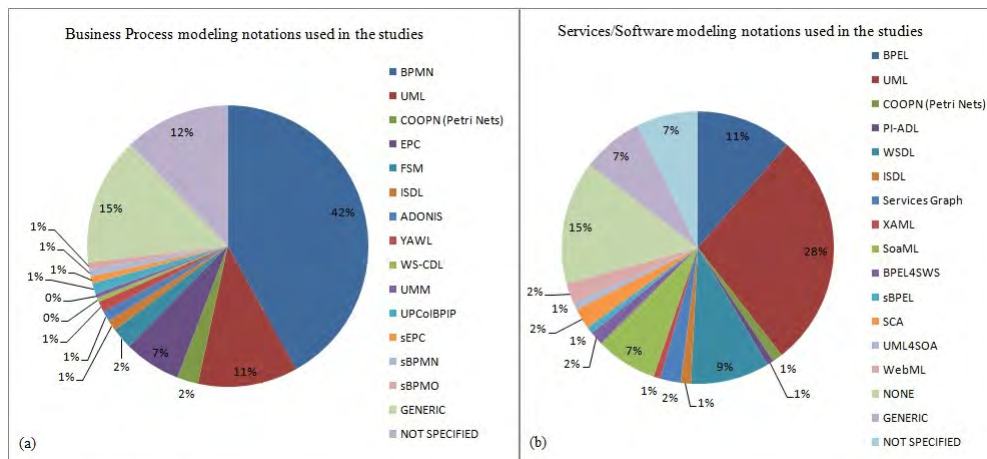


Figure 3.52.: Business Process (a) and Service/Software (b) notations used

Different notations can be used for each need identified ranging from business process modeling to execution, simulation, the assessment of desired and undesired properties for the models, specification of services, service interaction and composition. As can be seen in Figure 3.52 (a), the most widely used notation for business process modeling is BPMN (42%) and the most used notation for service/software modeling is UML (28%) shown in Figure 3.52 (b), the former being the main standard for business process modeling, since its adoption by the OMG in 2006, and the latter the accepted standard for software development. Other notations also used for business process modeling are UML (11%) and EPC (7%) (Figure 3.52 (a)), and for service modeling for business process execution, WS-BPEL (11%) and WSDL (9%) when service implementation is via WS, followed by SoaML (7%) (Figure 3.52 (b)).

In the majority of the studies, at least one notation is used or recommended for business process and service-oriented modeling, although generic notations are sometimes used or no specific notation is mentioned at all, usually when the approach is methodological and/or any notation could be used along with the proposal.

### 3.5.3.2. Main Principles in Paradigm Integration

In this section the main principles regarding SOC, MDD and BPM paradigm integration as found in the systematic review are presented and discussed along with an illustration from the studies. These principles are seven: Business Process modelling, Service Oriented modelling, Models transformations, Methodological approach, Use of patterns, Collaborative processes and Tool support. The presence of each principle in the total of the studies selected is shown in Figure 3.54.

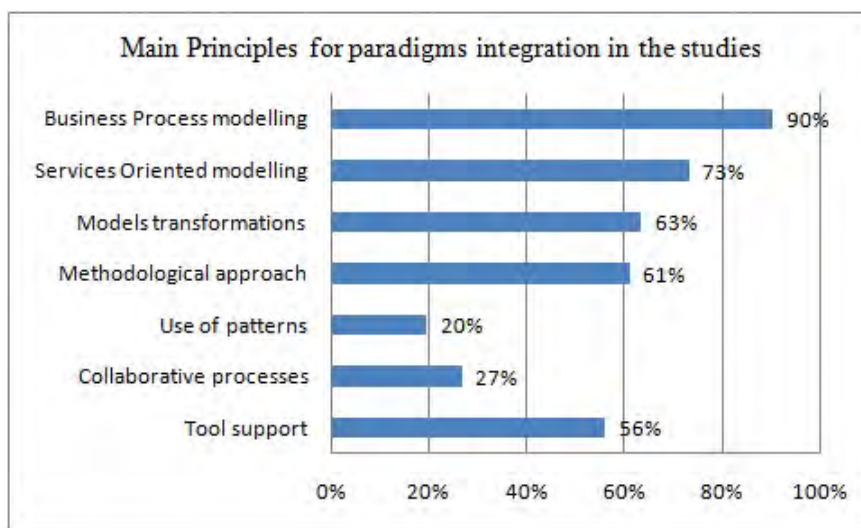


Figure 3.53.: Main principles in paradigms integration

Each sub-section corresponds to a main principle found, and the studies discussed in each one are the most relevant ones for each principle. The last sub-section presents a summary of principles by paradigms integration, including where the principles each study deals with are summarized.

#### Business Process Modeling

One of the most important issues for the support of business processes (BP) by service orientation (SO) in a model-driven way is how the organizations handle their business processes, especially regarding the explicit modeling in some notations, including the flow of the activities to be performed, the data exchanged and the roles involved. There are a variety of notations that can be used to specify business processes; each has advantages and disadvantages depending on the needs of the organization, the type of BP and the use intended for the model, as shown in Fig. 2(a).

There are many studies modeling BP using BPMN, and they include [Liew et al., 2004] where the BPMN BP model is annotated with extra information to be used in transformations. In [Tao Tao

and Yang, 2006] BPs are modeled after the analysis of services needed from system functionalities. [Henkel and Zdravkovic, 2005] model BP that are then realized by technical processes using existing services that match business functionality. [Gacitua-Decar and Pahl, 2008] uses an enhanced BPMN by domain model elements and an own UML profile. [Rychly and Weiss, 2008] models BP with BPMN then transformed into services diagram.

[Thomas and Leyking, 2008] define an initial conceptual model in EPC for process design transformed into a conceptual-technical application model in BPMN for process configuration. [Oquendo, 2008] use BPMN to model BP which are then transformed into a PI-ADL formal language. [Touzi et al., 2009] use BPMN to model collaborative BP adding a special pool called Collaborative Information System (CIS), which mediates between partnered information systems. In [Delgado et al., 2009c] BPMN is proposed to guide the service-oriented development from BP models and BP execution models which are then linked to each other, and in [Delgado et al., 2010b] BPMN models are transformed directly into service models.

In [Elvesaeter et al., 2010] and in [Dahman et al., 2010] BPMN models are also used to obtain service models. In [Bai and Wei, 2009] an adapted BPMN is used, and in [Brambilla et al., 2009] an extended BPMN, in both cases to include information about execution to generate the associated execution process. UML is also used as in [Mili et al., 2006] to represent and classify generic BP to be used in several domains which are then instantiated using a catalogue of software components. [Zdun et al., 2007] use AD to model BP as a long-running interruptible process called macroflow and IT-oriented processes as a short-running transactional process called microflow, where both layers defined are above a business application service layer.

[Quartel et al., 2005] uses AD to model BP mapping them into an ISDL conceptual model related to application design and implementation models via the ISDL models. [Herold et al., 2008] uses AD to model BP, and a use case diagram and business structure diagram to model other aspects of the business view. [Bruckmann and Grunh, 2008] uses AD to model BP representing functions with actions and process with activities. EPC is found in studies such as [Roser et al., 2006] to specify the BP from which to obtain services in UML, [Murzek et al., 2006] and [Mendling et al., 2006] who present horizontal transformations between different notations, the former between EPC and ADONIS and the latter between yEPC and YAWL.

Other less frequently used notations are Concurrent Object Oriented Petri Nets (COOPN) in [Chen and Buchs, 2006] to model BP in the design phase at a PIM level and FSM (Finite state machine) in [Tao Tao and Yang, 2007] to model core BP, specifying generic activities and separating others applicable to a particular usage context. In [Lazarte et al., 2010] the UML profile for Collaborative BP based on Interaction Protocols (UPColBPIP) is used to model collaborative process to obtain a BPMN representation of each participant process, and in [Norton, 2009] semantic extensions for BPMN and EPC (sBPMN, sEPC) are used to add semantic information to the models. Some generic notations are used in studies to show activities, data and the flow of the process, but with no specific notations.

### Service Oriented Modeling

Another important issue regarding service support for business processes is the service-oriented modelling approach. Once the BP are known and modeled, each defined task from the process and even the process itself have to be realized by one or a group of services. Although the flow of the execution of services can be automatically obtained from the BP model, the definition, design, model and implementation of services is of great importance for tracking their correspondence to BP and to existing or new systems providing them. Services and software modeling is mainly done using UML, which is shown in Fig. 2(b), as in [de Castro et al., 2006] where the first step is to define the needed services and from them obtain use cases, service processes to support them and then generate the required service composition.

[Roser et al., 2006] model services showing the definition of services obtained for each proposed architectural approach (centralized and decentralized broker and brokerless) from the CIM description of business, [Zdun et al., 2007] use AD at a microflow level, showing the services involved in the IT technical processes, [Gacitua-Decar and Pahl, 2008] categorize services into: business services abstracting activities on business entities, and technical services abstracting functionality



and data provided by applications and to manage issues such as security and messaging. Although the use of the UPMS profile for services is mentioned it is not shown in the study.

[Rychly and Weiss, 2008] define a service diagram using an UML profile, specifying interfaces providing only one functionality, port, service consumer and provider, along with sequence and composition service diagrams. [Herold et al., 2008] use stereotypes such as ServiceAction, which represents the services provided by the application layer with core functionality. [de Castro et al., 2008] models the IS view at PIM level, with business services to be offered by the system as uses cases, and functionalities and process needed, [Bruckmann and Grunh, 2008] defines software modeling with class diagrams and state diagrams.

[Touzi et al., 2009] defines three views for SOA model: services for business functionalities, information for data and messages exchanged between services, and process for services interaction. Other used notation are WS-BPEL and WSDL, as for example in [Thomas and Leyking, 2008] where WS-BPEL is derived for execution from BPMN models as in [Bai and Wei, 2009] among others, [Hu and Grefen, 2003] uses WSDL to describe services which is also used in other studies in conjunction with WS-BPEL, [Oquendo, 2008] uses a formal architectural language PI-ADL for SOA, including orchestration and choreography and generate WS-BPEL for process execution.

Other less used notations are as in [Quartel et al., 2005] where services are modeled using an ISDL dialect defining components that provides application services obtained from the business model, [Cauvet and Guzelian, 2008] where business services are modeled with three parts: profile (goal), structure (process) and process part (BP) and service composition using a service composite graph, or generic ones such as in [Tao Tao and Yang, 2006] where services are identified from essential functionalities of the system.

The new SoaML standard is used in [Delgado et al., 2010b] and in [Elvesaeter et al., 2010] to model services transformed from BPMN models, in [Lazarte et al., 2010] another services profile is used, the UML4SOA as an example of services modeling, and in [Dahman et al., 2010] the Service Component Architecture (SCA) is used from BPMN models too. In [Norton, 2009] a semantic extension sBPEL comprising BPEL and BPEL4SWS is defined to add semantic information to the models. BPEL4SWS is also used in [Weber et al., 2009] along with WSMO to describe services functionality and activity implementations.

### **Model Transformations**

Transformations between models used for the specification of BP and services are one key aspect of paradigm integration. Many approaches have been proposed to transform and generate software models from BP models, where existing languages can be used to define mappings and transformations, although new ones or different approaches are also defined. The OMG Query/Views/Transformations (QVT) [OMG, 2008c] standard and ATL [Jouault and Kurtev, 2005] are the most relevant examples. Transformations not only make it possible to automatically obtain elements of a target model from an origin model, but also to explicitly specify the correspondences between elements and the semantics involved.

Vertical transformations from one level of abstraction to another generally applied in a top-down way can be found in [de Castro et al., 2006] where four PIMs are defined to model system behavior: user services, extended use case, service process and service composition, defining mapping rules that can be completely or partially automated. In [Chen and Buchs, 2006] embedded process controllers are generated from BP to be integrated into existing IS such as ERP via service interfaces, generating java components. [Quartel et al., 2005] define transformations to travel from one ISDL conceptual model to another, from business to service implementation. [Mili et al., 2006] uses a question approach defining variation points and BP variants in generic BP which are then mapped to a software components library of generic business components, to automatically assemble software systems.

In [Roser et al., 2006] three different architectural approaches for software systems (centralized and decentralized broker and brokerless) are derived from a CIM description of business, establishing how services for each approach correspond to BP. [Zdun et al., 2007] apply transformations successively based on defined patterns, starting with the macroflowmicroflow pattern which establishes

the conceptual basis and the process-based integration architecture pattern that guides the design of an architecture based on sublayers for the service composition layer. [Henkel and Zdravkovic, 2005] propose going from BP models to technical processes matching existing services by applying transformation patterns classified with respect to the quality of the transformation.

In [Gacitua-Decar and Pahl, 2008] conceptual transformations are defined based on the successively application of patterns from the top to the bottom layer, using graphs for pattern matching. [Rychly and Weiss, 2008] define two steps for BP to services transformation: identifying tasks in BP representing service invocations, then using a proposed technique that integrates BP and object modeling into a Business Service Model (BSM), mediating between business requirements and implementation. [Herold et al., 2008] go from a business model (CIM) to an analysis model (PIM), identifying serviceAction in tasks, then to a Design model (Architecture specific model, ASM), mapping services to the target architecture where each component provides a set of services.

[de Castro et al., 2008] define transformations from a value model to use case models, defining mapping rules from the CIM to PIM level between model elements, which are automated using ATL and tools. [Bruckmann and Grunh, 2008] use stereotypes actions that map to user and system functions in a defined metamodel described in an XML schema as interchange format and input for transformation engines. [Oquendo, 2008] define mapping between BPM constructs and PI-ADL for SOA expressions, showing mappings for a subset of process patterns and BPMN core elements. [Touzi et al., 2009] define two types of transformation rules: basic generation to create elements of the target model, and binding rules to generate links between them, using ATL.

[Delgado et al., 2009c] propose transformations from BPMN to service models in SoaML from which to generate services implementation, and from BPMN to execution models in BPEL/XPDL using existing approaches, from which to invoke the generated services. In [Delgado et al., 2010b] a set of transformations from BPMN to SoaML service models is presented, which are defined using QVT. In [Elvesaeter et al., 2010] transformations from BPMN to SoaML are also proposed but using ATL and a different definition of mappings. [Dahman et al., 2010] also defines several mapping rules to generate SCA models from BPMN models which are implemented also using ATL.

[Brambilla et al., 2009] proposes a set of models and transformations to obtain an application executable model in WebML from an extended BPMN model, from which code in J2EE is generated. In [Lazarte et al., 2010] several transformations are defined between the models used going from collaborative BP to BPMN models representing the partners in the collaboration, and by intermediate models to obtain the code in the desired platform. In [Norton, 2009] a chain of transformations involving defined ontologies to navigate from sBPMN and sEPC models to BP MO models, which provides a common abstraction for BP modeling, from which to generate sBPEL models using WSML to effect this transformations.

Horizontal transformations on the same level of abstraction can be found in [Murzek et al., 2006] based on control flow patterns [van der Aalst et al., 2003a], identifies patterns in the original BP, transforming each one into the target notation to obtain the target BP model, and in [Mendling et al., 2006] where transformations are based on elements of each notation and the algorithm traverses the yEPC process graph node by node to transform the BP. In [Sadiq et al., 2006] an automatic distribution of collaborative BP from an integrated one is proposed, defining the correspondences and algorithms to extract the distribute models from the integrated one. In [Sinha and Paradkar, 2010] transformations are defined from use case models to BPMN models, synchronizing the requirements definition.

A combination of vertical and horizontal transformations can be found in [Liew et al., 2004] where the BPMN BP model is annotated with information processed by the defined algorithms, and then transformed into several UML software artifacts: horizontal from BP to AD, vertical from BP to use cases, collaboration and deployment diagrams. [Orriens et al., 2006] define mappings between defined models in three levels that can be horizontal or vertical; using five elements capturing particular facets: what, how, where, who and when. [Thomas and Leyking, 2008] define horizontal transformations between EPC conceptual model to BPMN conceptual-technical model, and vertical ones from BPMN to BPEL for process execution.

### Methodological Approach

When modeling business processes, services and other software artifacts needed to support software development, a systematic approach to guide the development is essential. Even if some artifacts can be obtained automatically from others, a guide for the activities to be done and the flow between them, among other aspects, are a key factor for success. Software development processes have been successfully used in recent years, such as Unified Process [Jacobson et al., 1999], and approaches to include service views, activities and artifacts to guide service development have also been defined.

[Papazoglou and van den Heuvel, 2006] define a methodology for SO design and development from business models, defining SO design and development principles such as service coupling, cohesion and granularity. It defines six phases: planning, analysis (process identification, process scoping, business gap analysis and process realization) and design (service design concerns, specification, business processes), construction and testing, provisioning (service governance, certification, metering and rating, billing strategies), deployment, execution and monitoring which are traversed iteratively.

[Kohlborn et al., 2009] after reviewing thirty existing service development approaches, propose a consolidated approach that combines examined methodologies and adds new items. They define two main parts for the process: the derivation of business services with four phases of preparation, identification, detailing and prioritization, and the derivation of software services to support them with phases: preparation, identification and detailing. [Tao Tao and Yang, 2006] based on the methodology [Papazoglou and van den Heuvel, 2006] adopts three primary phases of BP analysis, design and implementation, defining for BP analysis the steps: services identification, component identification, process scoping and process realization analysis.

[Zhao et al., 2006] define the following phases: contracting, collaboration and design to define services provided and required by each organization involved and to design and coordinate collaborations. [Herold et al., 2008] propose a model-driven approach with four phases: business development, requirement analysis, architectural design and implementation modeling, with guidelines and transformations to move from one model to the other. [de Castro et al., 2006] defines a method for service composition with a process comprising several steps related with model generation, defining metamodels, models and artifacts to be obtained from each step, specifying activities with tasks and inputs and outputs. The business model is the general input for the process and its output the services composition model.

[Gacitua-Decar and Pahl, 2008] define steps to apply pattern techniques to successively refine BP into services that realize them, identifying business patterns in BP, and technical services. [Thomas and Leyking, 2008] define three phases: process design, configuration and execution including in each the defined models and transformations. [de Castro et al., 2008] define an SODM service-oriented development method with an MDA-based approach with a process and transformations between models. [Touzi et al., 2009] define models, metamodels and transformations to go from collaborative BP (CIM) to the SOA model (PIM) from which to generate code (BPEL), based on the PIM4SOA.

[Delgado et al., 2009c] propose a methodology for service oriented development from business process defining disciplines, activities with objectives, input and output artifacts and associated roles, from BPMN models to service-oriented design and implementation. In [Bai and Wei, 2009] six steps are defined with activities to navigate from a BPMN model which is remodeled with execution information, to a BPEL implementation. In [Weber et al., 2009] a methodology is proposed for the modeling and configuration of BP adding a semantic approach to implement services from BP. In [Brambilla et al., 2009] a top-down model-driven approach is proposed defining several steps from going to BPMN models to implementation using WS and Web interfaces.

In [Patig and Wesenberg, 2009] an hybrid service design process is proposed which defines modeling BP from which to identify services in a top-down manner and also a bottom-up identification from information concepts and existing application systems. For B2B development, in [Baghdadi, 2004] a design process is proposed, consisting of six steps: BP specification, decomposition and distribution specification, mapping and validation, supporting services and components specification, logical

B2B application architecture, implementation and integration technology, defining objective, input and output artifacts and models and tools. [Hu and Grefen, 2003] define steps for going from BP definitions, implementing activities that can be internal applications or remote services provided by others, with a service mediating layer to bridge activity specifications with its implementation.

[Huemer et al., 2008] define a top down methodology based on existing approaches, starting from business and BP models to services deployment artifacts, outlining a description, notations and tools to use for each step, backed with a software factory to generate code. In [Lazarte et al., 2010] a top-down model-driven approach is proposed with four phases: Business Analysis, Design of Business solution, Design of IT Architecture solution and Design of the Technological solution, defining activities, roles, artifacts and transformations. Other approaches include different visions for service development, as in [Cauvet and Guzelian, 2008] which defines an iterative service composition process in which services matching BP requirements are selected and alternative services can be generated, using ontologies to match BP requirements to service goals.

[Chen, 2008] define the BITAM-SOA framework for business-IT alignment extending ATAM, via architecture, governance and communication, defining three layers comprising specific modules, using them to guide a process model for service design, development and management, which can be top-down or bottom-up. In [Tao Tao and Yang, 2007] four steps in service development are defined based on core BP determination, adding a usage context creation for Configurable Context BP (CCBP) generation and service interface derivation.

### Use of Patterns

Design patterns for software development are well known and have been used by the community for a long time, with the reference point being that of GoF [Gamma et al., 1995]. For business process modeling, the most relevant work is [van der Aalst et al., 2003a] in which several BP constructions are defined and analyzed. The importance of reusing the best existing solutions for known problems is well established in the Software Engineering literature, so another key aspect for paradigm integration is the use of patterns at various stages of development.

Different pattern approaches are presented, as in [Zdun et al., 2007] where a pattern language with patterns and primitive patterns is defined for the integration of BP and technical processes based on services, which are applied successively in a top-down way from a macroflow defined process. In [Henkel and Zdravkovic, 2005] transformation patterns are defined to be used when going from BP to technical processes, applying levels of realization and realization types of BP by using existing services to match BP in lossfull, constrained, lossless and exceeded realizable transformations. [Gacitua-Decar and Pahl, 2008] define business patterns in two types: process and domain patterns and SOA patterns; a pattern catalogue organizes them into templates.

In [Elvesaeter et al., 2010] a process fragment pattern in BPMN is defined to transform it to a service contract in SoaML. In [Lazarte et al., 2010] predefined activity patterns are used to refactoring a BPMN interface model into a BPMN integration model. [Murzek et al., 2006] use workflow patterns [van der Aalst et al., 2003a] for the control flow aspect of BP models, as a basis for the horizontal transformations between different BP notations. [Oquendo, 2008] also uses process patterns to map BPMN constructs to PI-ADL expressions which are iterative and applied to the original BP for transformations to services in PI-ADL.

Process patterns are also used in [Norton, 2009] in the definition of a graph-oriented common abstraction of BP in the BPMO ontology as part of the SUPER project. In [Roser et al., 2006] the Broker architectural pattern is used and patterns for service interaction are modeled in UML collaborations. Other architectural patterns are also mentioned in various studies, being the most used the Layers pattern to define and organize architectural levels.

### Collaborative Processes

The modeling of collaborative processes adds complexity and coordination requirements to business process models, as the involved participants has to agree on the points and ways of interaction with each other. However, collaborative process modeling is one of the most needed activities in organizations, in order to best define the way to perform their collaborative business with their

partners in a coordinate and beneficial way, and to be able to evaluate requirements for changing their BPs.

[Orriens et al., 2006] present a Business Collaboration Context Framework (BCCF), capturing models in a business collaboration information model (BCIM) with three levels: strategic, operational and service level, with mappings which are then developed and managed, driven by rules which make it possible to validate and verify the alignment between model elements. In [Roser et al., 2006] SO systems realizing collaborative BP are derived from a business level in a model and architecture-driven development perspective (architecture approaches centralized and decentralized broker and brokerless) as part of the ATHENA project, where the PIM4SOA comprising a set of metamodels and tools allows the description of services and their collaborations at PIM level.

[Zhao et al., 2006] provide support for BP collaborations of dynamic virtual organizations on the basis of a service-oriented relative workflow, adding definitions for IT and BP, where private information is hidden by wrapping local workflows into perceivable workflows according to visibility constraints for defined perceptions. [Touzi et al., 2009] propose a methodology for developing collaborative architectures following the MDA approach from collaborative BP to the SOA model, adding specific elements for collaborative modelling as an intermediate collaborative pool in BPMN models, and collaborative services from it.

In [Delgado et al., 2010b] collaborative BP are modeled in BPMN to be transformed into services models in SoaML using pools, activities and messages flows. In [Dahman et al., 2010] BPMN conversations are used to model participants and messages exchanged to be transformed into SCA models. For B2B application development, [Baghdadi, 2004] defines a four-layered architecture with four interrelated abstraction levels: business models and process, BP decomposition and distribution, supporting services and integration technology to guide the design process. [Hu and Grefen, 2003] define a three-level conceptual framework and architecture to flexible service enactment in B2B collaborative processes, with a service-mediating layer to bridge the BP definition with its implementation with services.

[Huemer et al., 2008] define three layers based on the Open-edi reference model for inter-organizational systems: business operational view (BOV) comprising business models and BP model layers, and the functional service view (FSV) comprising a deployment artifacts layer. In [Lazarte et al., 2010] a global view of B2B process is modeled from which the BP model corresponding to each participant is obtained and progressively transformed to the code for the B2B specification of the partner's integration BP and system interfaces in the desired technology.

### **Tool Support**

Another key aspect for paradigm integration is tool support for each development stage in the business and software development effort. To effectively help closing the business and systems gap, the use of tools that enable a smooth integration between both areas, models and artifacts is needed.

Studies that provide their own tool support include [Orriens et al., 2006] with the tool ICARUS developed to support the framework proposal, [Sadiq et al., 2006] with a BP editor that implements the algorithms for BP distribution and [Zdun et al., 2007] with a model-driven tool chain that supports modelling and model validation, using UML transformed into DSL syntax, validating models and transforming them into EMF for generating Java and BPEL code. [Oquendo, 2008] uses an own core toolset and customizable tools for PI-ADL developed previously for the Archware project. In [Bai and Wei, 2009] an own tool implementing the BPMN remodeling metamodel is presented as snapshots but no description of the tool is given. [Elvesaeter et al., 2010] use the CIMFlex editor tool developed in the SHAPE project to BP modeling including business rules and data, for EPC and BPMN models transforming to SoaML models using ATL.

Other studies use existing tools, adding their own support when needed, as in [Chen and Buchs, 2006] where an IDE COOPN Builder is used to edit, visualize and verify COOPN modules, including a java and WS generator. [de Castro et al., 2006] use the Oracle BPEL development tool and own tools developed for the MIDAS framework, while in [Quartel et al., 2005] an ISDL editor

and simulator is used adding a prototype of a BPEL profile for ISDL. [Mili et al., 2006] use EMF to define metamodels and the Eclipse BPEL plug-in to model BP. [Tao Tao and Yang, 2006] use the Oracle BPEL process manager for implementation and EJBs, and apacheAxis and MySQL as infrastructure.

[Roser et al., 2006] use ARIS for BP modelling and a prototype implemented in ATHENA for the generation of services and BPEL processes, [Hu and Grefen, 2003] use the IBM MQSeries workflow as a basis for a prototype implementation of the proposal. [Huemer et al., 2008] evaluate the MS DSL Tools for Visual Studio and ADONIS as candidate tools for the software factory. [de Castro et al., 2008] use Eclipse with EMF for metamodel definition, GMF for model visualization and ATL for transformations, working on code generation into different WS platforms. In the same direction, [Touzi et al., 2009] use a set of Eclipse tools: Intalio designer for collaborative BP modeling, EMF for metamodel definition, ATL for transformations and TOPCASEDO for visualizing UML models. In [Dahman et al., 2010] Eclipse with EMF for metamodeling is also used, as is ATLAS for implementing the chain of transformations defined.

[Delgado et al., 2009c, 2010b] also use EMF for metamodel definition, integrating several Eclipse plug-ins to provide support for the defined phases and activities including BPMN and SoaML modeling and QVT transformations with the MediniQVT plug-in. [Lazarte et al., 2010] also propose the use of Eclipse integrating existing plug-ins for modeling with UPColBPIP, BPMN, BPEL and WSDL, and others for transformations based on ATL, QVT, VIATRA, JET2. [Brambilla et al., 2009] use an existing tool WebRatio supporting WebML design and code generation for web applications, which they extended to add BPMN extensions, the BPMN to WebML transformation and the generation of code to J2EE to include the new primitives. References for the tools mentioned can be found in the studies.

### 3.5.3.3. Summary of Main Principles and Selected Studies

The main principles we have found in the selected studies for the integration of paradigms are also related to the paradigms the studies integrate. Figure 3.54 shows the main principles by paradigms integration in the studies.

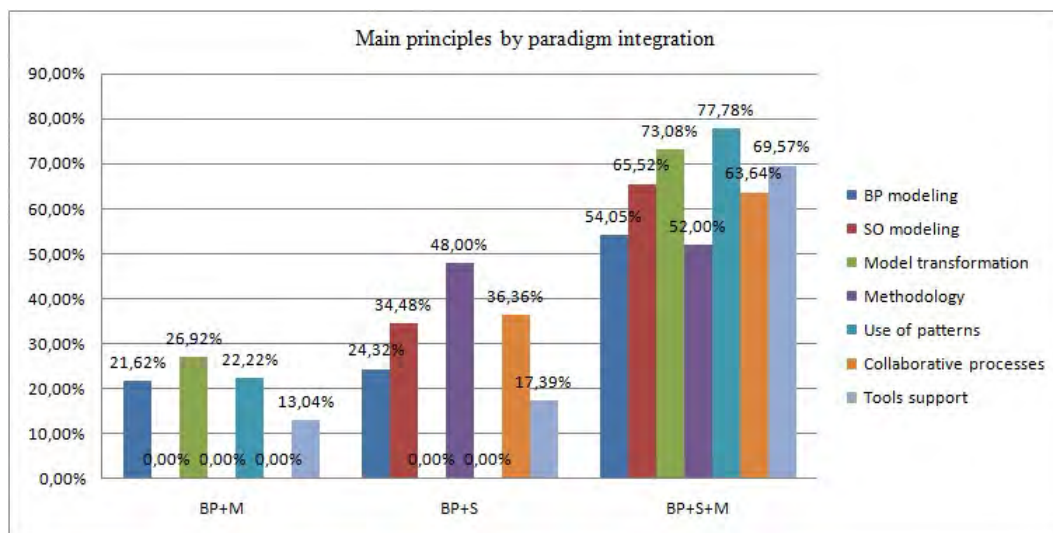


Figure 3.54.: Main principles in selected studies by paradigms integration

In the category BP+S+M all principles are present, as expected, as in general the studies belonging to this category propose complete integration of paradigms for navigating from business process models to software design and implementation with services or not, in a model-driven way based on a defined methodology and using existing or new proposed patterns, with existing or their own developed tool support. Business process modeling and tool support are the only principles that are present in the three categories for paradigm integration defined: BP+S+M, BP+S and BP+M.

This can be seen as a clear demonstration of the importance that business process modeling is gaining in software development, and that tool support is increasingly needed for the different phases and activities it comprises.

As paradigms integration drive the definition of categories, it is consistent to see that in the BP+S category there are no model transformations proposed (when they are, then the category of the study is BP+S+M) and in the BP+M category there is no service-oriented modeling but software modeling (in general in the form of UML classes, interactions, etc.). For the BP+S category there is no use of patterns in the proposals, they are associated mainly with the model-driven paradigm, in which of course it is a basic principle defined by the approach, but we believe they should also be used in any software development approach, as they provide several benefits such as reuse of solutions, quality of the software developed, among others.

It can be seen that for the BP+M category there are no methodological approach mentioned in the studies (only informal steps in a general model-driven procedure) and no reference to collaborative processes. A plausible interpretation is that in a model-driven approach, the approach itself is used as a guide for development, and the complexity added by collaborative processes makes them more difficult to manage in model transformations than in conceptual or methodological proposals.

Business process and service-oriented modeling have proven to be two of the most important principles regarding the relation between business and software areas. To move from one area to another, two other principles were found to provide the basis for bringing the areas closer: the methodological and the model-driven approach. The former provides guidelines to develop services systematically from business processes, while the latter, by means of mappings, rules and transformations, makes the automation of development possible, with models and metamodels being the key elements.

Two important principles to improve the quality of the solutions are: the definition or reuse of patterns for BP and service-oriented modeling which helps reduce errors in early stages of development, and the tool support for development, including facilities for the verification of model properties, simulation of process and execution of transformations. Patterns can be used in a conceptual way to guide the modeling effort or the derivation of elements of a target model from an origin model, and also to automate this derivation, by the execution of defined model transformations.

**Table 3.4.:** Summary of primary studies selected and main principles found in each study

Author/Year	Paradigm integration	BP modeling	SO modelling	Model transf.	Methodology	Patterns use	Collab. process	Tool support	Case study*
(Hu et. al, 2003)	BP+S	NO	YES	NO	YES	NO	YES	YES	E
(Baghdadi, Y. 2004)	BP+S	YES	YES	NO	YES	NO	YES	NO	NO
(Liew et al, 2004)	BP+M	YES	NO	YES	NO	NO	NO	NO	E
(Henkel et al, 2005)	BP+M	YES	NO	CP	NO	YES	NO	NO	E
(Quartel et al, 2005)	BP+S+M	YES	YES	YES	NO	NO	NO	YES	E
(Chen et al, 2006)	BP+S+M	YES	YES	YES	NO	NO	NO	YES	NO
(de Castro et al, 2006)	BP+S+M	NO	YES	YES	YES	NO	NO	YES	R
(Mendling et al, 2006)	BP+M	YES	NO	YES	NO	NO	NO	NO	E
(Mili et al, 2006)	BP+M	YES	NO	YES	NO	NO	NO	YES	E
(Murzek et al, 2006)	BP+M	YES	NO	YES	NO	YES	NO	NO	E
(Orriens et al, 2006)	BP+S+M	YES	YES	YES	NO	NO	YES	YES	R
(Papazoglou et al, 2006)	BP+S	YES	YES	NO	YES	NO	NO	NO	NO
(Roser et al, 2006)	BP+S+M	YES	YES	YES	NO	YES	YES	YES	E
(Sadiq et al, 2006)	BP+M	YES	NO	YES	NO	NO	NO	YES	NO

Author/Year	Paradigm integration	BP modelling	SO modelling	Model transf.	Methodology	Patterns use	Collab. process	Tool support	Case study*
(Tao Tao et al, 2006)	BP+S	YES	YES	NO	YES	NO	NO	YES	R
(Zhao et al, 2006)	BP+S	YES	YES	NO	YES	NO	YES	NO	E
(Tao Tao et al, 2007)	BP+S+M	YES	NO	YES	YES	NO	NO	NO	E
(Zdun et al, 2007)	BP+S+M	YES	YES	CP	YES	YES	NO	YES	R
(Bruckmann et. al, 2008)	BP+M	YES	NO	YES	NO	NO	NO	NO	NO
(Cauvet et. al, 2008)	BP+S	YES	YES	NO	YES	NO	NO	NO	NO
(Chen, H., 2008)	BP+S	NO	YES	NO	YES	NO	NO	NO	E
(de Castro et al, 2008)	BP+S+M	YES	YES	YES	YES	NO	NO	YES	E
(Gacitua-Decar et al, 2008)	BP+S+M	YES	YES	CP	YES	YES	NO	NO	NO
(Herold et. al, 2008)	BP+S+M	YES	YES	YES	YES	NO	NO	NO	E
(Huemer et. al, 2008)	BP+S	YES	YES	NO	YES	NO	YES	YES	NO
(Oquendo, F., 2008)	BP+S+M	YES	YES	YES	NO	YES	NO	YES	E
(Rychly et. al, 2008)	BP+S+M	YES	YES	YES	YES	NO	NO	NO	E
(Thomas et. al, 2008)	BP+S+M	YES	YES	YES	YES	NO	NO	NO	E
(Bai et al., 2009)	BP+S	YES	YES	NO	YES	NO	NO	YES	E
(Brambilla et al., 2009)	BP+S+M	YES	YES	YES	YES	NO	NO	YES	R
(Delgado et al., 2009)	BP+S+M	YES	YES	YES	YES	NO	YES	YES	NO
(Kohlborn et. al, 2009)	BP+S	NO	YES	NO	YES	NO	NO	NO	E
(Norton et al., 2009)	BP+S+M	YES	NO	YES	NO	YES	NO	YES	NO
(Patig et al., 2009)	BP+S	YES	NO	NO	YES	NO	NO	NO	R
(Touzi et. al, 2009)	BP+S+M	YES	YES	YES	YES	NO	YES	YES	E
(Weber et al., 2009)	BP+S	YES	NO	NO	YES	NO	NO	NO	NO
(Dahman et al., 2010)	BP+S+M	YES	YES	YES	YES	NO	YES	YES	E
(Delgado et al., 2010)	BP+S+M	YES	YES	YES	NO	NO	YES	YES	E
(Elvesaeter et al., 2010)	BP+S+M	YES	YES	YES	NO	YES	NO	YES	E
(Lazarte et al., 2010)	BP+S+M	YES	YES	YES	YES	YES	YES	YES	E
(Sinha et al., 2010)	BP+M	YES	NO	YES	NO	NO	NO	YES	E

\* E = Example, R = Real

### 3.6. Conclusions

In this Chapter the review of the state of the art for the main research subjects of this thesis has been presented: Business Process Management (BPM), Service Oriented Computing (SOC), Model Driven Development (MDD) and Continuous Business Improvement (CPI). Several definitions, concepts and key elements involved in each of the subjects studied were presented, which make up the basis of the definitions in MINERVA framework, along with main standards for each one, that are integrated in MINERVA. Paradigms were put in historical context, showing the evolution each one presented in the last two decades or less, and providing the vision of the current situation in terms of advances, languages, standardization, and use.

In addition, a systematic literature review carried out at the beginning of this research work on the application of SOC and MDD to BPM and BPs has been presented, along with the results obtained. Seven main principles that have to be taken into account for the integration of paradigms SOC, MDD and BPM were found as the main result of the systematic review: business process modeling, service oriented modeling, model transformations, methodological approach, use of patterns, collaborative processes and tool support, which have been taken into account in the definition of MINERVA.



The reasonable man adapts himself to the world: the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.

---

*George Bernard Shaw*

## Chapter 4.

# MINERVA framework

This Chapter describes the “Model driven and service oriented framework for the continuous improvement of business process & related tools” (MINERVA) which is the central work of this thesis, and provides an integrated and standardized approach to support the management and improvement of BPs realized by services with a model-driven approach.

The description of the proposal is organized as follows: in section 4.1 the motivation behind the search for an integrated framework to support the continuous improvement of BPs realized by services is presented, and in section 4.2 the general definition of MINERVA framework is described. MINERVA’s main elements are set out based on two views: Dimensions presented in section 4.3 and Process presented in section 4.4. Finally, in section 4.5 conclusions for the chapter are discussed.

The contents of this chapter are complemented by those of the following chapters: chapter 5, which presents the Business Process Continuous Improvement Process (BPCIP) defining the complete BP lifecycle from modeling to improving the BPs, chapter 6, which describes the Business Process Execution Measurement Model (BPEMM) defining BP and service execution measures to guide the measurement effort within the defined lifecycle, chapter 7, which presents the Business Process Service Oriented Methodology (BPSOM) to guide the development of service-oriented systems from BPs with a model-driven approach, chapter 8, which describes the MDA approach to generate SoaML service models from BPMN2 models, chapter 9, which presents the tools support defined for the work throughout the MINERVA framework, and chapter 10, which presents the validation of MINERVA framework carried out by means of two case studies and an experiment designed to validate the defined transformations.

### 4.1. Motivation

BPM [Weske, 2007] is being increasingly adopted by organizations wanting to gain insight and control into their BPs, so as to be able to react to changes, and to improve them continuously; to do that several aspects have to be taken into account if a BPM project is to be carried out in an appropriate way. These factors include support for the BP lifecycle, methodologies and technologies to implement the BPs, and ways to find and integrate improvement opportunities. Several elements, steps and tools for applying and supporting the BPM paradigm effectively are needed, along with a guide to carry out a BPM project in a systematic way, thereby guaranteeing the successful of the effort undertaken. To meet all these requirements, MINERVA framework has the following characteristics:

- The realization of BPs by means of services applying the Service Oriented Paradigm (SOC) [Papazoglou et al., 2007] which provides the basis for separating their definition from the technologies that implement them. It helps provide a better response to changes in either of the layers defined -i.e. definition and implementation of BPs- with minimum impact on the other. Services can implement an activity, a sub-process or a complete BP, and can be integrated easily into the BP execution without the interoperability problems that had to be solved formerly for the systems to achieve integration. This approach helps close the so called business-systems gap, originated by different views and expectations between the business and IT areas when introducing changes into the BP models and their implementation [Krafzig

et al., 2005, Erl, 2005]. A systematic approach to deriving services from BPs helps guide the development process.

- Model transformations for the automatic generation of service-oriented models from BP models, to ease the design of services and to support the effective separation between definition and implementation of BPs. Models have proven to play an important role in the software development process: informally to sketch out the concepts of the system for evaluation of solutions and communication among stakeholders; or in a more formal way specifying them without ambiguity using metamodels, models and languages to allow transformations between models, this being the basis for Model Driven Development (MDD) [Mellor et al., 2003, Schmidt, 2006, Karakostas and Zorgios, 2008]. One of its key uses in the context of BP realization by means of services is that of designing services at a more abstract level than with specific technologies, and promoting traceability among defined elements [Karakostas and Zorgios, 2008], as well as generation of code in different technologies.
- “You can’t manage (to improve) what you can’t measure” is an old business principle defining the importance of measurement for improvement. An improvement effort has to support the identification of process deficiencies and provide guidance for introducing improvements in a systematic way. To that end measures of the BP, activities, performance, resources, cost and results have to be defined, implemented, collected and analyzed on a regular basis. It is not enough to provide measures and the means to analyze them, including tool support, it is also essential to align measures with business strategy and business goals for the entire organization, with the ones that are specific to each BP. This will allow to interpret the information collected from their execution correctly. The primary focus should be first on understanding and defining BPs business goals (i.e. performance, results, costs, etc.) and then measuring their business results against these goals [OMG, 2008b].

MINERVA framework covers a wide range of elements included in the BPM, SOC and MDD research areas, for which several different subjects have been studied and evaluated. The definition of MINERVA framework has been carried out as completely as possible to cover all the phases in the BP lifecycle adding new elements to give explicit support to the continuous improvement of BPs. It should be said, however, that due to the wide variety of topics that this integration requires, some elements have been either left out from this thesis, as they are beyond its scope, or only outlined as possible solutions and left held pending for future work.

## 4.2. MINERVA definition

MINERVA [Delgado et al., 2009c, 2010h] is a framework for business process improvement based on the application of the SOC and MDD paradigms to BPs and BPM, guided by the BP lifecycle [Weske, 2007] and the use and implementation of existing standards from OMG, WfMC and OASIS. It provides an integral vision for the development of service-oriented systems from BPs with a model-driven approach, and guidance for the continuous improvement of BPs based on BPs execution measures including an execution measurement model with a set of pre-defined execution measures. By “framework” we mean a set of different kind of elements in order to provide systematic support for a certain goal, in our case the integrated use of SOC, MDD and BPM elements for the continuous improvement of BPs.

MINERVA can be explained from two different points of view: Dimensions and Process. The Dimensions view, for its part, organizes elements in three different dimensions: conceptual [Delgado et al., 2010f], methodological Delgado et al. [2009b, 2011c, 2010b, 2012a, 2011e] and tool support [Delgado et al., 2010a], defining the structural view of the framework. The Process view sets out the lifecycle and method of work that will guide the work throughout the framework, based on the elements defined in the Dimensions view, providing the dynamic view of the framework. In Figure 4.1 the two views defined in MINERVA are presented.

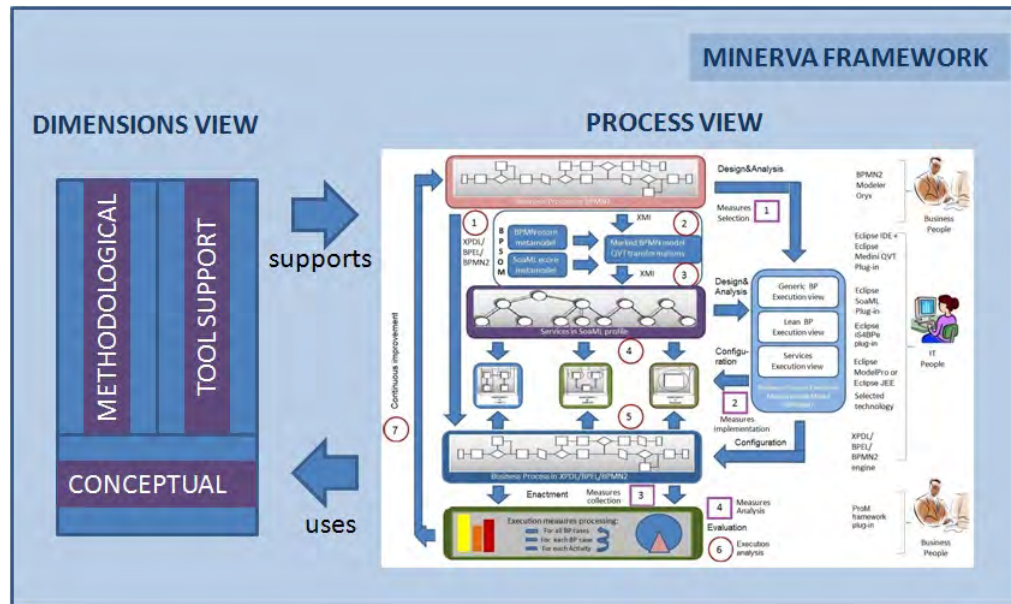


Figure 4.1.: MINERVA framework views

From the Dimensions point of view MINERVA defines three kinds of elements:

- The Conceptual dimension defines the concepts and the relationships between them, needed to support the BP lifecycle, which will be further used throughout the framework. This is the basis upon which the other two rely, similarly to how the Infrastructure packages are defined in metamodel standards such as UML. It defines a conceptual architecture, a set of metamodels and an ontology made up by seven sub-ontologies to support the BP lifecycle.
- The Methodological dimension provides the approaches (methods and techniques) to be used. A process to guide the continuous improvement of BPs and an execution measurement model which incorporates execution measures to guide the measurement effort. A previously defined service oriented methodology has been extended to guide the implementation of BPs by means of services, including a model-driven approach for the automatic generation of service oriented models in SoaML from BP models in BPMN2.
- The Tools Support dimension integrates existing and newly developed tools to support the work in each of the phases and activities defined by the MINERVA lifecycle using the elements defined previously. The focus of this dimension is to provide a main integrated environment based on the Eclipse platform, extended with a selected set of existing plug-ins along with newly developed ones. It also integrates external tools to support activities that are carried out by business people, such as BP modeling and execution measurement evaluation. The main criterion for the selection of tools was for them to be free or at least freeware, as defined by GNU<sup>1</sup>, our own contribution to the community was made by releasing the Eclipse SoaML plug-in that had been developed in this thesis.

The Process view provides the guide for the effective use of the elements defined in the Dimensions view, which are used throughout the lifecycle and method of work that has been defined to guide the work throughout the framework. The lifecycle of MINERVA is defined by the Business Process Continuous Improvement Process (BPCIP) included in the Methodological dimension, which consist of the BP lifecycle [Weske, 2007] extended with explicit measurement activities, as well as improvement activities based on the PmCompetisoft [Pino et al., 2009]. The process view defines the phases and method of work throughout these, where a key element is the explicit modeling of BPs in BPMN2, guiding the rest of the phases and activities defined. The focus is on the organization and in the business people as those who have the business knowledge that needs to be taken into account for the systems support to be effective.

<sup>1</sup><http://www.gnu.org/philosophy/categories.html>

### 4.3. Dimensions view

The Dimensions view provides the structural view of the MINERVA framework, defining the elements to be used throughout the framework as well as the relationships between them. The elements defined in each dimension: Conceptual, Methodological and Tool support, are presented below.

#### 4.3.1. Conceptual dimension

The Conceptual dimension defines the base elements to be used in MINERVA framework, as shown in Figure 4.2, and includes:

- A **conceptual Architecture** for the framework which includes a particular instantiation of the four-layered Architecture defined by the Meta Object Facility (MOF) [OMG, 2011b] standard from the OMG, to help manage the complexity of metamodeling and modeling between these different levels.
- A **collection of metamodels** needed to support the model-driven approach defined by the framework, including the domain metamodels for the BP modeling with BPMN2 and service-oriented modeling with SoaML and UML, and the metamodel for defining model transformations with the QVT language.
- An **ontology** to support the BP lifecycle realized by services, which is composed of seven sub-ontologies defining the main concepts and relationships for each of the phases defined, as well as for BPs and the services realizing them.

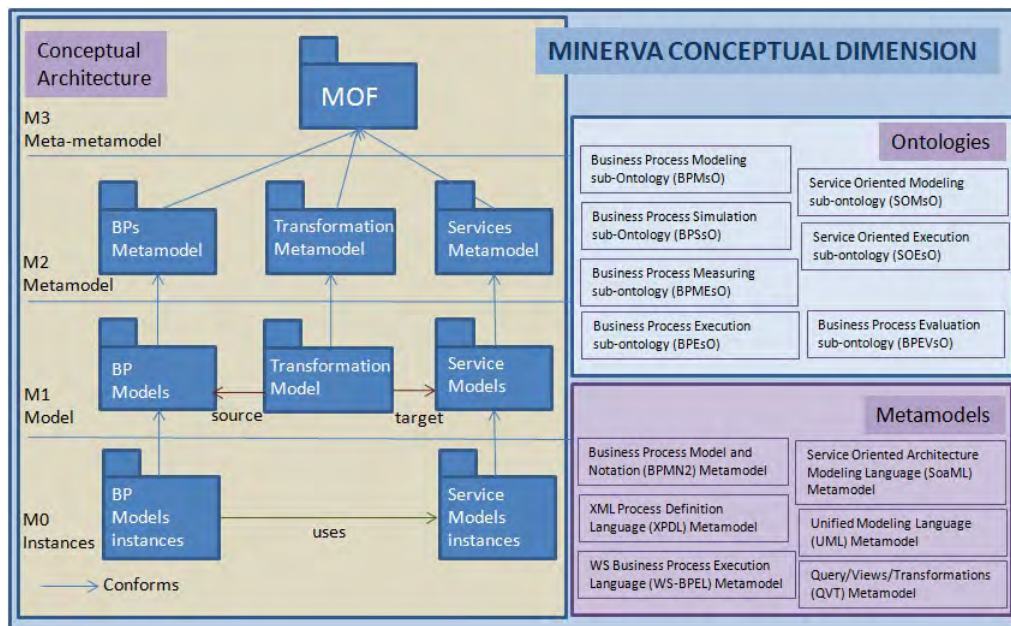


Figure 4.2.: Conceptual Dimension elements

##### 4.3.1.1. Conceptual Architecture

The Conceptual Architecture provides the definition of the levels of abstraction needed to manage the different metamodels and models that are used in MINERVA. This architecture shows the integration of metamodels and models for BPs and services, as well as for the defined transformations between them (which are described in chapter 8). The levels of abstraction are described below, as presented in chapter 3, showing the particular instantiation for MINERVA.

- **M3 - Meta-metamodel level:**

It is at this level that MOF itself resides, which allows metamodels for different domains to be determined based on the elements it defines. The EMF ECORE from Eclipse is similar to MOF (EMOF) which enables metamodels to be manipulated in the Eclipse environment, which is the one used in MINERVA framework.

- **M2 - Metamodel level:**

At this level metamodels are specified by means of MOF, in the case of MINERVA we include metamodels for BP specification, which can be for modeling or execution, metamodels for services specification, as well as metamodels for defining the transformations to generate service models from BP models. All metamodels at this level must be MOF compliant. Although the conceptual approach for defining these metamodels is general, in MINERVA framework we use only the metamodels we have integrated.

- **M1 - Model level:**

At this level models for different domains are specified based on the metamodels defined in the upper M2 level. These models are constructed by means of the elements defined by the corresponding metamodel, as well as the restrictions that it imposes over the elements. All the models at this level conforms to the specification of the corresponding metamodel from which it constitutes a specific instance. Thus, a BP model showing a process from a Hospital or a Bank, for example, will use the same elements from the metamodel to specify the model, although the meaning of the model will represent different domains or different situations in the same domain.

- **M0 - Instances level:**

At this level several instances of the same model defined in the previous M1 level occur, i.e. the specific objects and data from the real world corresponding to the elements modeled in the previous level, are found here. These instances conform to the definitions in the associated model specified at the previous level. For example, in the case of the BP model showing a process in a Hospital, the specific occurrence of the process for Patient A, and the specific occurrence for Patient B, possibly with a different illness and result. Data from the occurrence of these instances is used to evaluate the execution of BPs and services (which is described in chapter 5).

#### 4.3.1.2. Metamodels

Several metamodels to be used for the modeling of BPs and services are included in MINERVA, to define the transformations between BPs and services, as well as in the execution of BPs and services. The metamodels we have integrated in MINERVA are mentioned briefly below along with their use, as there is a complete description in chapter 3.

- **Business Process Model and Notation (BPMN2):**

BPMN2 [OMG, 2011a] integrates a notation, a metamodel, an exchange format and the complete semantics for the execution of BPMN2 models. Using the BPMN2 metamodel BPs models can be specified and executed, and exchanged between modeling tools and process engines to execute them. These facilities are used throughout the different phases of MINERVA to model BPs in BPMN2, export the models and load them into the Eclipse environment, as well as to generate SoAML services from BPMN2 models, insert services invocation into BPMN2 models, and execute BPMN2 models invoking the services generated.

- **XML Process Definition Language (XPDL):**

XPDL [WfMC, 2008] defines another exchange format for BP models that can be executed in workflow engines. This metamodel is used in MINERVA framework to provide another option for the execution of BPs, integrating existing facilities for the conversion of BPMN2 models into XPDL models to be deployed in the process engines using workflows, as well as to insert services invocation into the XPDL model corresponding to the services generated.

- **WS Business Process Execution Language (WS-BPEL):**

WS-BPEL [OASIS, 2007] is the standard for the execution of BPs in Web Services engines. This metamodel is used in MINERVA framework to provide another option for the execution of BPs, integrating existing facilities for the conversion of BPMN2 models into WS-BPEL models to be deployed in process engines using WS, as well as insert services invocation into the WS-BPEL model corresponding to the services generated.

- **Service Oriented Architecture Modeling Language (SoaML):**

SoaML [OMG, 2009b] is a profile and metamodel extending the UML metamodel, providing specific stereotypes for services modeling. This metamodel is used in MINERVA framework for the generation of SoaML service models from BPMN2 models, as well as for the visualization of the models generated in the Eclipse SoaML plug-in we have developed (and in the creation of SoaML service models from scratch without automatic generation).

- **Unified Modeling Language (UML):**

UML [OMG, 2011c] metamodel is used as the basis for the SoaML extension, as the SoaML metamodel extends and specializes several elements defined in UML. For example, the ServicesArchitecture and ServiceContract elements from SoaML are UML Collaborations, so inheriting the properties and constraints defined for collaborations in UML. For some definitions in SoaML, moreover, there are no specific stereotypes or elements defined, rather they use specific elements directly from UML (i.e. a choreography for a ServiceContract can be specified using any interaction diagram: sequence, collaboration, activity diagram).

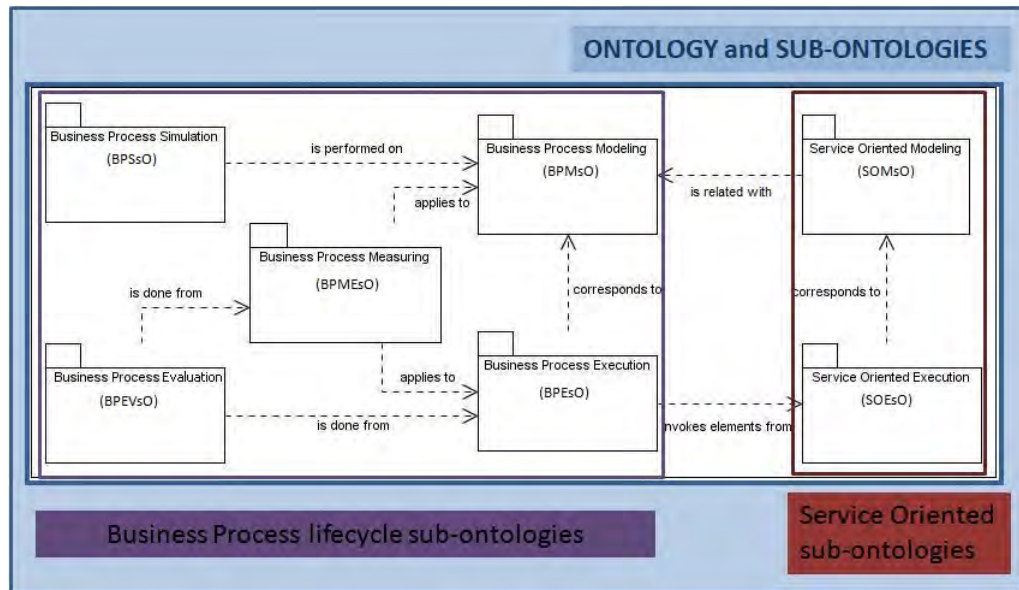
- **Query/Views/Transformations (QVT):**

The QVT [OMG, 2008c] metamodel is used to model the transformations for generating SoaML services from BPMN2 models. It is used in the MINERVA Design environment along with the BPMN2 and SoaML metamodels to define the rules for the transformations, which are then executed in a QVT engine we have integrated. The philosophy of representing Model to Model transformations as models is followed, as presented in chapter 3.

#### 4.3.1.3. Ontologies

The ontologies defined in MINERVA framework have the purpose of identifying the concepts which will underpin the manipulation of the elements needed, ensuring a consistent definition which takes into account the relationships that exist between the elements in different conceptual areas (for example between BPs and the services realizing them). It enables there to be definition, organization and reuse of knowledge about concepts involved in the BP lifecycle, as well as their design and implementation based on services. An ontology defines relevant elements (concepts, relationships) in a given area of interest [Gruber, 1993, 2009], providing meaning to the vocabulary and formalizing restrictions on its use [Gruber, 2009]. Agreeing on the terminology used help us identify which elements of the reality are important for the metamodels and models representing such reality, filtering out non-necessary knowledge.

Based on the BPs lifecycle [Weske, 2007], five conceptual groups have been identified to define sub-ontologies for: modeling, simulation, execution, measurement and evaluation of BPs, respectively. For service orientation, two conceptual groups have been identified, defining sub-ontologies for service-oriented modeling and execution. The ontology specified can be classified according to the "Taxonomy of ontologies for software engineering and software technology" [Ruiz and Hilera, 2006] corresponding to item A) Ontology of domain/Software technology (ST)/ Information Technology and Systems / Models and Principles. The seven sub-ontologies comprising the high level ontology are shown in Figure 4.3.



**Figure 4.3.:** Ontology and sub-ontologies for BP lifecycle realized by services

In Figure 4.3 the sub-ontologies and their relationships are shown; on the left side the ones related to the BPs lifecycle and on the right the service-oriented ones. Horizontally, it can be observed, that the Service Oriented Modeling sub-Ontology (SOMsO) is related to the Business Process Modeling sub-Ontology (BPMsO), and the Service Oriented Execution sub-Ontology (SOEsO) is “used” by the Business Process Execution sub-Ontology (BPEsO), meaning that BPEsO “uses” elements from the SOEsO, where, in an execution of a BP the execution of services realizing it will be invoked. Vertically, the BPEsO corresponds to the BPMsO, meaning that the elements from the second one trace to elements from the first, the same occurs for the SOEsO which corresponds to the SOMsO.

The three remaining sub-ontologies correspond to concepts which support the BPs lifecycle by defining: the Business Process Measuring sub-ontology (BPMEsO) which integrates measures for BP models and execution [Sánchez González et al., 2009]; the Business Process Evaluation sub-ontology (BPEVsO) which “uses” elements from measuring and execution sub-ontologies, defining other elements used to analyze execution, for example for event logs analysis; the Business Process Simulation sub-ontology (BPSsO) defines elements to simulate and understand various characteristics of models prior to their execution.

We have detailed the sub-ontologies corresponding to the BP and service modeling, since we have used them to reason about the elements and relationships between them, as the basis for the definition of the transformations between the corresponding metamodels. The rest of the sub-ontologies have only been sketched out; this is just to provide a general view of the elements and relationships involved -detailed specification of these is left for future work.

In the following, the BP and services modeling sub-ontologies are described, and an overview of the rest of the sub-ontologies, indicating the sources that have been evaluated for each definition. Our aim when defining the ontologies was to obtain simple ones which were nevertheless capable of expressing the most relevant concepts and relationships in the conceptual groups, without filling them with details based on specific definitions of BPs or services.

### **Business Process Modeling sub-Ontology (BPMsO)**

For the definition of the Business Process Modeling sub-Ontology (BPMsO) the concepts needed to express a BP model have to be defined and related in a way that reflects how the elements involved can be combined. By the time the BPMsO sub-ontology was defined, the BPMN2 standard was under revision by the OMG and was only released at the beginning of the year 2011, which is why we have evaluated the previous version of BPMN2 the Business Process Modeling Notation (BPMN) v.2.1 [OMG, 2009a] together with the metamodel in the Business Process Metamodel Definition (BPDM) [OMG, 2008a].

In spite of the different versions core elements of the BPMN standards remained unchanged throughout them, so the BPMsO could still be used as the basis for the mappings between BPMN2 and SoaML when transformations were defined. As the concepts defined in these standards covers, in our understanding, the relevant elements for business process modeling, we have mainly adopted them directly, also adding new ones. In the following the BPMsO is presented in two tables around the levels defined in the sub-ontology, indicating where the term is a new one, or the source from where it was either Adopted as is, or Adapted mainly merging definitions from the sources, with our own vision of the concept. The Term column indicates the term being defined, the Level column indicates a hierarchy defined between the concepts (1 being the highest level), the Source column indicates BPDM as source one (S1) and BPMN as source two (S2) and the Definition column provides the description of the concept.

**Table 4.1.:** Definition of terms in the first levels of BPMsO

Term	Level	Source	Definition
Business Process	1	New	Real process in the organization
BPMModel	2	New	Business process model that specifies a real BP.
BPMModel Notation	3	New	Valid notation to specify a business process model
BPType	3	New	Indicates the type of business process
Orchestration	4	Adopted S1	Sequence of activities that produces results with branching and merging. Describes what happens and when in order for a process to be managed under the authority of an specific entity
Choreography	4	Adopted S1	Describes how semi-independent collaborating entities work together in a process, each of one has its own internal processes. Captures in a given process the roles interaction with well defined responsibilities.
BPMModel Element	3	New	Elements in a business process model

In Table 4.1 the concepts defined in the first levels of the sub-ontology, around the element BPMModel representing business process models are shown. The BPMModel has a type which can be orchestration or choreography, it is specified in a valid notation, such as BPMN, and it is composed of elements for business process modeling derived from BPMModelElement, which are shown next in Table 4.2.

**Table 4.2.:** Definition of terms within the BPMModelElement of BPMsO

Term	Level	Source	Definition
Artifact	4	Adopted S2	Provides information support about the process or their elements, without affecting the process flow
Data	5	Adopted S2	Provides information about what activities require to be performed and what produce
Group	5	Adopted S2	Represents an activity group that are in the same category
Swimlane	4	Adopted S2	Container to partition a set of activities from others
Pool	5	Adopted S2	Represents a process participant that is a business entity (enterprise, section) or a business role (seller, buyer) that controls it.
Lane	5	Adopted S2	A sub-partition of a Pool used to organize and categorize activities
Connecting Object	4	Adopted S2	Object used to connect two objects with each other showing how the flow goes



Term	Level	Source	Definition
Sequence	5	Adopted S2	Shows the order in which process activities are performed, from start to end
Message	5	Adopted S2	Shows the message flow between two participants prepared to send and receive
Association	5	Adopted S2	Associates information to flow objects, could be text or graphical objects not of flow
FlowObject	4	Adopted S2	Objects flow could be: activities, events, gateways
Activity	5	Adopted S2	Generic term for the work that an organization performs
SubProcess	6	Adapted S1+S2	Activity that is composed of other activities, process included in another process.
Simple	6	Adopted S1	Activity which is no composed of other activities
Gateway	5	Adopted S2	Decision point used to control the divergence and convergence of process flow
Parallel (AND)	6	Adopted S2	Flows are performed concurrently instead of sequentially, when diverging (fork) and converging (join)
Exclusive (XOR)	6	Adopted S2	Flow is restricted to only one alternative from the possible when diverging and combining (merging)
Inclusive (OR)	6	Adopted S2	Branching where different alternatives are evaluated and one, several or all of them can be selected, and for combining (synchronize)
Complex	6	Adopted S2	An expression determines which options are selected when diverging and which are required when combining (merge)
Event	4	Adopted S2	Something that happens in the course of a process affecting its flow, and generally has a cause (trigger) or an impact (result)
Flow Dimension	5	Adopted S2	Event that affects the process flow, can be start, intermediate or end
Start	6	Adopted S2	Event indicating where a process begins
Intermediate	6	Adopted S2	Event that happens between a start and an end event
End	6	Adopted S2	Event indicating where a process ends
Type Dimension	5	Adopted S2	Type associated to the event, can be message, time, error, cancellation, conditional, signal, compensation, link, termination or multiple
Actor	5	Adopted S1	Entity responsible for the execution of tasks specified by a realizing role .

Table 4.2 presents the concepts defined as business process elements within the `BPMModelElement`, allowing business processes to be represented as models in `BPMModel`. The `BPMModelElements` are grouped into `Artifacts`, `Swimlanes`, `FlowObjects` and `ConnectingObjects`, which are adopted mainly from BPMN and include concepts such as `Pool`, `Lane`, `Activity`, `Sub-process`, different types of `Gateways`, `Sequence` and `Message flow`, among others. The elements defined in these groups are then related to the service oriented modeling elements defined, described in the next section. In Figure 4.4 a UML class diagram for the `BPMsO` is shown to describe its structure.

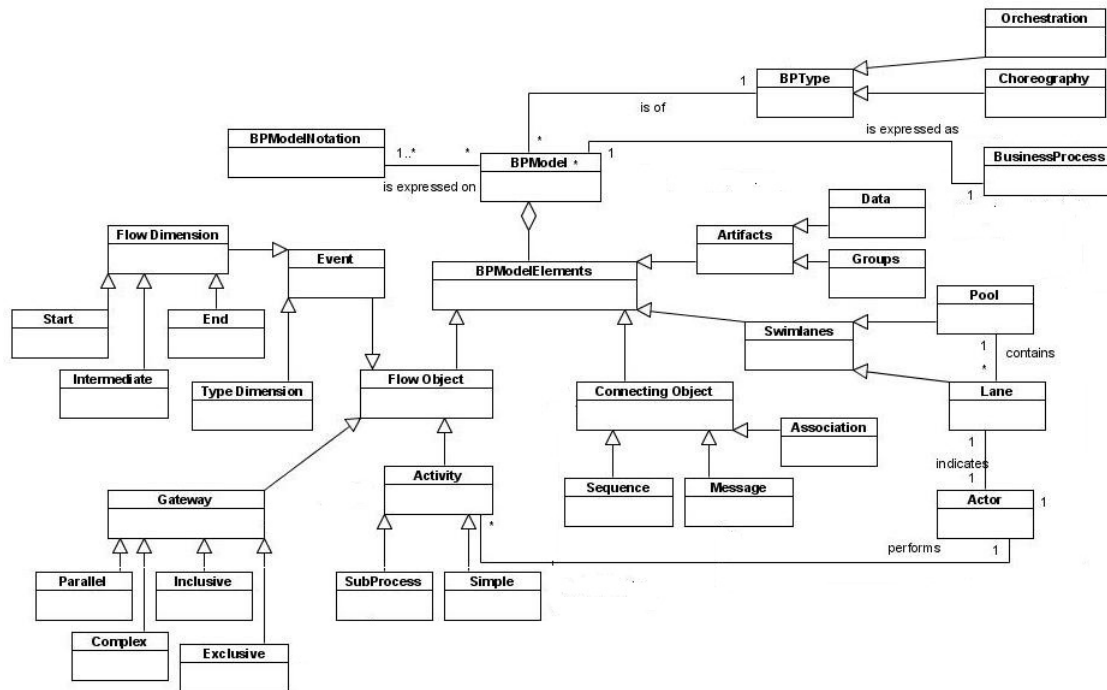


Figure 4.4.: BP Modeling sub-Ontology (BPMsO) diagram

### Services Oriented Modeling sub-Ontology (SOMsO)

For the definition of the Service Oriented Modeling sub-ontology (SOMsO), several existing international standards which establish ontologies, models and metamodels to describe services were evaluated. The most relevant ones are: Service Oriented Architecture Modeling Language (SoaML) [OMG, 2009c] from OMG, SOA Reference Model (SOA-RM) [OASIS, 2006] and SOA Reference Architecture (SOA-RA) [OASIS, 2008] from the Organization for the Advancement of Structured Information Standards (OASIS), SOA Ontology (SOA-O) [OpenGroup, 2008] from the Open Group and the Web Services Architecture (WSA) [W3C, 2004] from the World Wide Web Consortium (W3C).

As the concepts defined in these standards and their definitions are heterogeneous, we have conducted a comparison analysis between concepts and their definitions, with the goal of either to adopt a term definition from a source we considered most suitable for the interpretation in the context of MINERVA framework, or to adapt it merging definitions from several sources with our own vision, or adding a new term and definition. Although by the time the ontology was defined the SoaML standard was in a previous version beta 1, the changes between the two versions referred mostly to names of the stereotypes and not the concepts themselves, so the SOMsO could be used as the basis for the mappings between BPMN2 and SoaML when transformations were defined.

The Service Oriented Modeling sub-Ontology (SOMsO) is presented in three tables set out below, displaying the levels defined and the main terms, including the new ones and the reference to the source from which the term was either Adopted as it is, or Adapted mainly by merging definitions from more than one source, with our own vision of the concept. The Term column indicates the term being defined, the Level column refers to a hierarchy defined between the concepts (1 being the highest level), the Source column shows the sources: SoaML as source three (S3), SOA-RM as source four (S4), SOA-RA as source five (S5), SOA-O as source six (S6) and WSA as source seven (S7); the Definition column provides the description of the concept.

**Table 4.3.:** Definition of terms in the first levels of SOMsO

Term	Level	Source	Definition
Business Process	1	New	Real process in the organization
SOModel	2	New	Service oriented model to implement a business process model that specifies a BP
SOModel Notation	3	New	New Valid notation to specify a service oriented model
SOType	1	New	Indicates the type of services composition
Orchestration	2	Adapted S6+S7	Composition of services that defines the sequence and conditions in which one service invokes other services, directing and controlling them, in order to realize some useful business function.
Choreography	2	Adapted S3+S5+S6	Composition of services that are autonomous but have defined pattern of behavior with respect to each other, in order to achieve a common business goal. It defines what happens between provider and consumer without defining their internal processes, which have to be compatible with their service contracts.
SOModel Element	2	New	Elements in a service oriented model

Table 4.3 presents the concepts defined in the first levels of the sub-ontology, around the element Service Oriented Model, which has a type that can be orchestration or choreography depending on the composition of services defined, it is specified by means of a valid notation, such as SoAML, and is composed of elements for service oriented modeling, which are presented next in Table 4.4.

**Table 4.4.:** Definition of terms around Service term of SOMsO

Term	Level	Source	Definition
Service	3	Adopted S6	A logical representation of a repeatable business activity that has a specified outcome (e.g. check customer credit). It is self-contained, may be composed of other services, and is a "black box" to its consumers. A service has a provider, may have multiple consumers, and produces one or more effects (which have value to the consumers).
Interface	3	Adapted S3+S7	Abstract boundary that a service exposes. It is composed of operations that defines the types of messages (including parameters, exceptions, etc) involved in interacting with the service. A ServiceInterface is the means for specifying how to interact with a Service.
Contract	3	Adopted S7	A service semantics is the contract between the provider entity and the requester entity concerning the effects and requirements pertaining to the use of a service. The semantics of a service is the behavior expected when interacting with the service. The semantics expresses a contract (not necessarily a legal contract) between the provider entity and the requester entity.
Implementation	3	New	Software implementation of a service in a specific technology (JEE, .NET, WS)

Term	Level	Source	Definition
Port	3	Adopted S3	From UMLSuperstructure: A port is a property of a classifier that specifies a distinct interaction point between that classifier and its environment or between the (behavior of the) classifier and its internal parts. A Port may specify the services a classifier provides (offers) to its environment as well as the services that a classifier expects (requires) of its environment.
RequestPoint (Request)	4	Adopted S3	Defines the port through which a Participant makes requests and uses or consumes services.
ServicePoint (Service)	4	Adopted S3	Defines the connection point of interaction on a Participant where a service is actually provided or consumed
Operation	3	Adopted S3	A service Operation is any Operation of an Interface provided or required by a Service or Request Point. An Operation has parameters defining its inputs and outputs, preconditions and post-conditions, may raise Exceptions
Message	3	Adapted S3+S5	Medium of interaction between service participants. A MessageType is the specification of information exchanged between service consumers and providers. This information consists of data passed into and/or returned from the invocation of an operation or event signal defined in a service interface.
Interaction	3	Adopted UML	Behavioral unit focus in the observable information exchange between elements. Their most visible aspects are the messages between the lifelines.
Sequence Diagram	4	Adopted UML	Describes an Interaction by focusing on the sequence of Messages that are exchanged, along with their Occurrence Specifications on the Lifelines
Communica- tion Diagram	4	Adopted UML	Focus on the interaction between Lifelines where the architecture of the internal structure and how this corresponds with the message passing is central. The sequencing of Messages is given through a sequence numbering scheme.

Table 4.4 presents the concepts defined around the Service element, which is the main concept in service oriented modeling. A service is composed of and has associated with it several model elements, such as interface, contract, operations, implementation provided on a port, among others. Service interface/s provide(s) operations defined for the service, including type and format of messages to be exchanged. Contracts offer interfaces adding other elements for agreement in the use of the service. Another important concept in service oriented modeling is that of Participant; it determines the services providers and consumers, which are described next in Table 4.5.

**Table 4.5.:** Definition of terms around Participant term of SOMsO

Term	Level	Source	Definition
Participant	3	Adopted S3	The type of a provider and/or consumer of services. In the business domain a participant may be a person, organization or system. In the systems domain a participant may be a system, or component.
Actor	4	Adopted S6	Someone or something that does something. An actor can be a person or an organization or a piece of technology.
Agent	5	Adopted S5	Any entity that is capable of acting on behalf of a person or organization. In order for people to be able to offer, consume and otherwise participate in services, they require the use of an agent capable of directly interacting with electronic communications – a service agent.

Term	Level	Source	Definition
Consumer	6	Adopted S6	A consumer of a service (or some other thing) is an actor that uses it. A service can have one or more consumers.
Provider	6	Adopted S6	A provider of an activity (service) is an actor that takes responsibility for it being carried out. Every service has a provider.
NoAgent	5	Adapted S6	Any other kind of actor that is not an Agent actor.

In Table 4.5 concepts related to the Participant element which offers and consumes services, are presented. The Participant is specialized in several concepts such as Actor which may be an Agent or a NoAgent (person, organization), and where providers and consumers offer services in points of Port type (Service and Request) associated with the service implementation. In Figure 4.5 a UML class diagram for the SOMsO is shown.

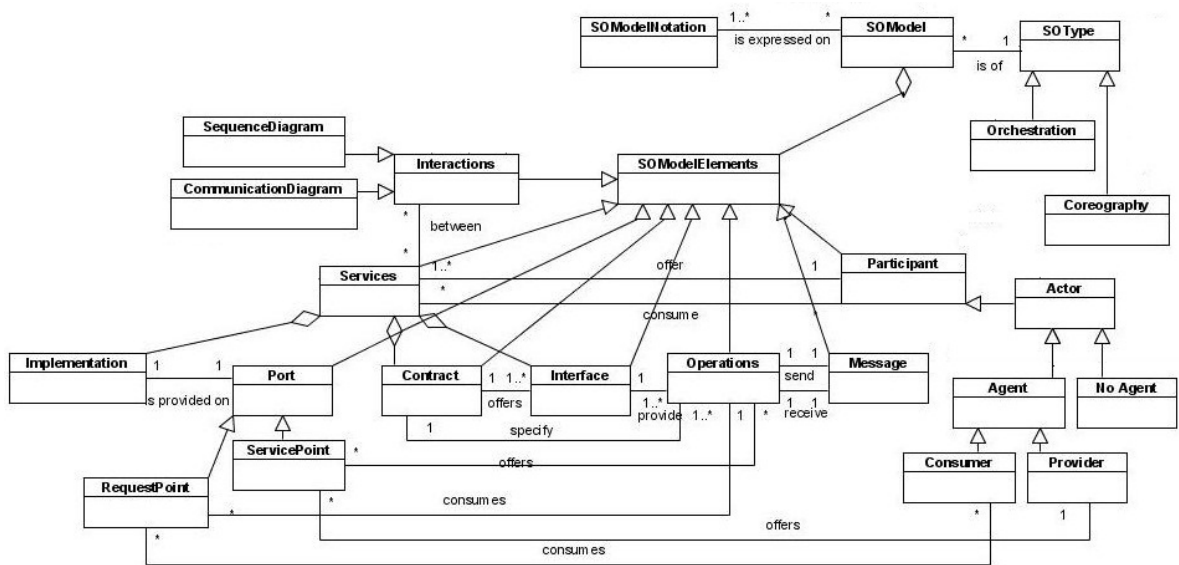


Figure 4.5.: Service Oriented Modeling sub-Ontology (SOMsO) diagram

### Business Process Simulation sub-Ontology (BPSsO)

The Business Process Simulation sub-Ontology (BPSsO) defines concepts for the simulation of BP models providing scenarios for execution of BPs depending on parameters that can be specified. The sub-ontology has only be outlined to identify the main concepts and relationships involved, based on the work in [Wynn et al., 2007]. Due to this no formal definition of concepts and relationships was established, and only a UML class diagram for the BPSsO was created to show the definition in diagram form, as presented in Figure 4.6.

The concepts and relationships to describe BP simulation are shown, based on a BPSimulation-Model that is composed of BPSimulationElements, including SimulationParameters that have to be defined for the simulation. DerivationFunctions can be specified using historical data extracted from previous logs (Simulation,Execution). Among these parameters TaskExecutionTime or Cost can be defined, or Breakpoints to stop simulation and to evaluate state. SimulationResults are given in SimulationReports and SimulationLogs, and BreakingState can also be register to be used as StartState in another simulation.

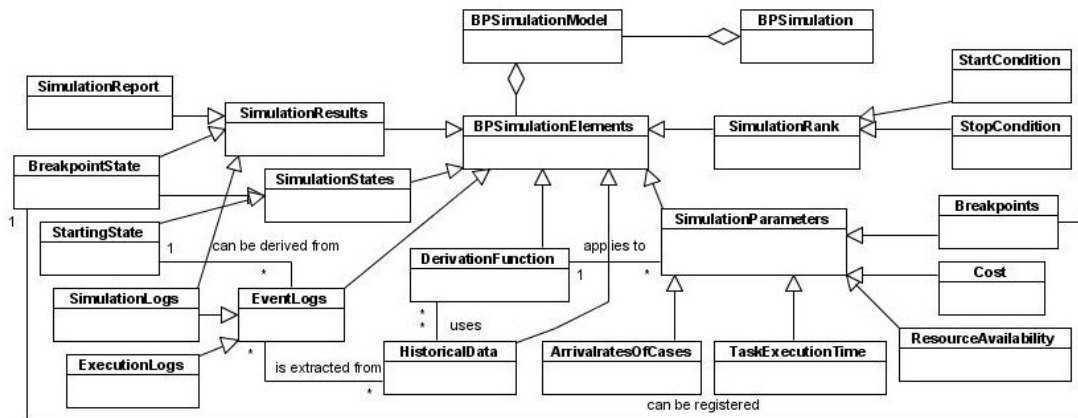


Figure 4.6.: BP Simulation sub-Ontology (BPSsO) diagram

This sub-ontology serves as a conceptualization of the elements involved in the simulation of BPs and was used to reason about what is needed to support its realization. We have not entered into details regarding simulation, or about other techniques for the validation and verification of BP model characteristics and have left this for future work.

### Business Process Measuring sub-Ontology (BPMEsO)

For the Business Process Measuring sub-Ontology (BPMEsO) we have integrated an existing measurement ontology, the Software Measurement Ontology (SMO) [García et al., 2005] that defines concepts and relationships to define and describe software measures. Whereas a BP model can be viewed as a conceptual artifact as is a requirements document, we can apply the SMO directly to measure them.

The SMO covered an extensive evaluation of standards such as IEEE Std. 1061-1998 (Software Quality Metrics Methodology), ISO/IEC 15939 (SE-Software measurement process), among others. A complete description of the SMO is out of scope, in Table 4.6 we present only some of its relevant elements, some of which we used for the definition of the execution measures in the BPEMM for BPs and services, as described in chapter 6.

Table 4.6.: Software Measurement Ontology (SMO) concepts from [Garcia et al., 2009]

Concept	SuperCon	Definition
Information Need	Concept	Insight necessary to manage objectives, goals, risks, and problems
Measurable Concept	Concept	Abstract relationship between attributes of entities and information needs
Entity	Concept	Object that is to be characterized by measuring its attributes
Entity Class	Concept	The collection of all entities that satisfy a given Predicate.
Attribute	Concept	A measurable physical or abstract property of an entity, that is shared by all the entities of an entity class
Quality Model	Concept	The set of measurable concepts and the relationships between them which provide the basis for specifying quality requirements and evaluating the quality of the entities of a given entity class
Measure	Concept	The defined measurement approach and the measurement scale (a measurement approach is either a measurement method, a measurement function or an analysis model)
Scale	Concept	A set of values with defined properties
Type of Scale	Concept	The nature of the relationship between values on the scale
Unit of Measurement	Concept	Particular quantity, defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity

Concept	SuperCon	Definition
Base Measure	Measure	A measure of an attribute that does not depend upon any other measure, and whose measurement approach is a measurement method
Derived Measure	Measure	A measure that is derived from other base or derived measures, using a measurement function as measurement approach
Indicator	Measure	A measure that is derived from other measures using an analysis model as measurement approach
Measurement Method	Measurement Approach	Logical sequence of operations, described generically, used in quantifying an attribute with respect to a speci- ed scale. (A measurement method is the measurement approach that de- nes a base measure)
Measurement Function	Measurement Approach	An algorithm or calculation performed to combine two or more base or derived measures. (A measurement function is the measurement approach that de- nes a derived measure)
Analysis Model	Measurement Approach	Algorithm or calculation combining one or more measures with associ- ated decision criteria. (An analysis model is the measurement approach that de- nes an indicator)
Decision Criteria	Concept	Thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result
Measurement Approach	Concept	Sequence of operations aimed at determining the value of a measurement result. (A measurement approach is either a measurement method, a measurement function or an analysis model)
Measurement	Concept	A set of operations whose objective is to determine the value of a measurement result, for a given attribute of an entity, using a measurement approach
Measurement result	Concept	The number or category assigned to an attribute of an entity as a result of a measurement

### Business Process Execution sub-Ontology (BPEsO)

To define the Business Process Execution sub-Ontology (BPEsO) we use the integrated metamodel [Hornung et al., 2006] of XML Process Definition Language (XPDL) [WfMC, 2008] from the Workflow Management Coalition (WfMC) and the Web Services Business Process Execution Language (WS-BPEL) [OASIS, 2007] from OASIS, covering concepts defined in both metamodels, as at the time the ontology was defined, these were the two major standards for process execution [Hornung et al., 2006].

The BPMN standard did not include semantics for execution until version 2.0, so it was not evaluated for this sub-ontology. The sub-ontology has only be outlined to identify the main concepts and relationships involved, for this reason no formal definition of concepts and relationships was established, and only a UML class diagram for the BPEsO was created to show the definition in diagram form, as presented inFigure 4.7.

The concepts and relationships defined for business process execution are shown, based on both XPDL and BPEL metamodels, defining concepts such as Process, Pool, Lane, Participant, Application, Activity which can be structured or basic and each of these types can also be specialized, classified in several types namely Handler for different type of events that can occur, Link and Message Flow, among others. The concepts for BP execution are closely related to those of modeling, and currently with the BPMN2 standard being executable, they are even the same if both modeling and execution are based on BPMN2. As several process engines implement the BPMN2 standard for execution, this will probably be the general case in the near future, so the BPEsO will also include the BPMsO.

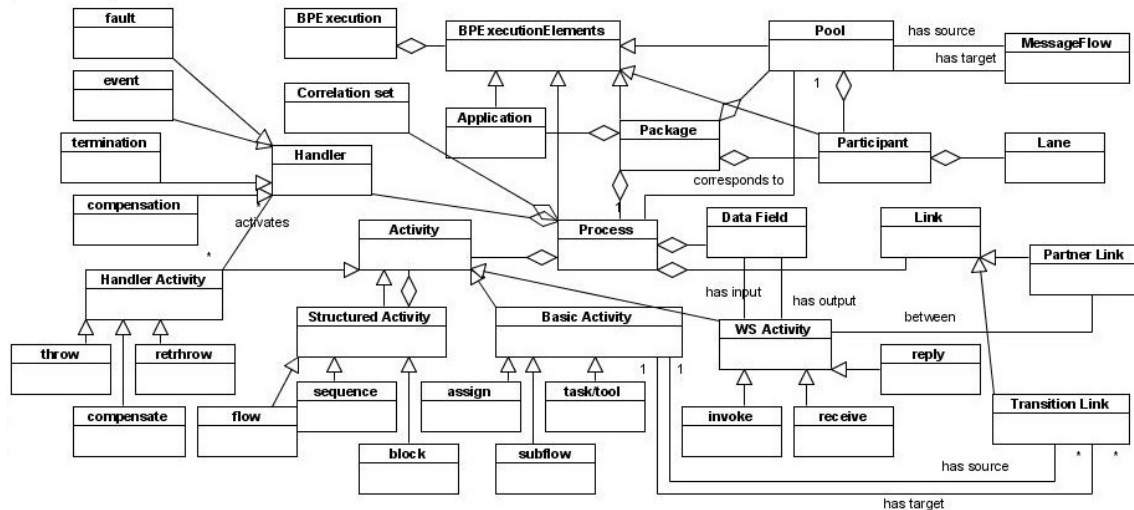


Figure 4.7.: BP Execution sub-Ontology (BPEsO) diagram

This sub-ontology serves as a conceptualization of the elements involved in the execution of BPs and has been used to reason about what is needed to support its realization. We have not gone into more detail for the execution of BP models and left it as future work.

### Service Oriented Execution sub-Ontology (SOEsO)

The Service Oriented Execution sub-Ontology (SOEsO) determines the elements that are required to describe the execution of services. Although it is directly related to existing technologies for service implementation, such as Web Services (WS), JEE, among others, there are shared concepts between them. Based on these concepts and the components definitions in UML standard from OMG the SOEsO was defined. The sub-ontology has only been outlined to identify the main concepts and relationships involved, for this reason no formal definition of concepts and relationships was produced, and only a UML class diagram for the SOEsO was created to show the definition in diagram form, as presented in Figure 4.8.

The main concepts and relationships for service oriented execution defined include SOExecution representing the service oriented execution obtained from the SOModel, which has a Technology-Type, and is composed of SOExecutionElements. The SOExecutionElements include terms such as Component, Interface, Port, Class, among others. Regarding the main relationships between elements these are that a Component is composed of other Components and/or of Classes, has Ports that can be provided or requested, which are associated with the interfaces provided and requested, that are used or implemented by the Component.

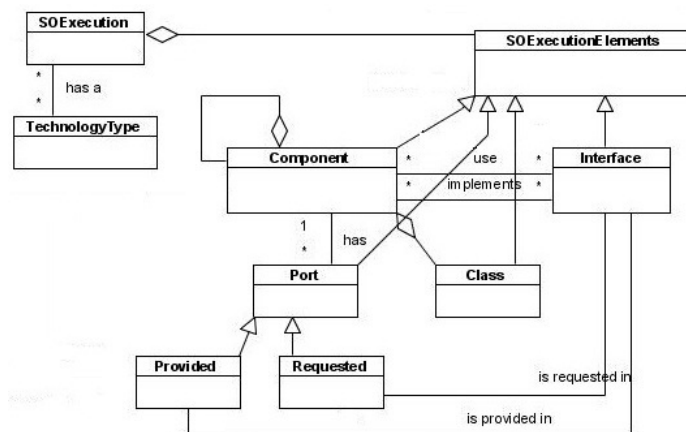


Figure 4.8.: Service Oriented Execution sub-Ontology (SOEsO) diagram



### Business Process Evaluation sub-Ontology (BPEVsO)

For the definition of the Business Process Evaluation sub-Ontology (BPEVsO) we based the identification of concepts on the MXML<sup>2</sup> format for workflow logs used by the ProM<sup>3</sup> framework [?], as presented in chapter 3. The aim of this sub-ontology is to identify the concepts and relationships needed to evaluate the execution of BPs. The sub-ontology has only been outlined to identify the main concepts and relationships involved, for this reason no formal definition of concepts and relationships was established, and only a UML class diagram for the BPSsO was created to show the definition in diagram form, as presented in Figure 4.9.

The most relevant concepts and relationships defined for the BPEVsO include the EventLog concept which defines each of the information row that will be stored, registering the Process and ProcessInstance occurrences to which the log is associated, and for each instance registering which Event had occurred, listed in the associated type EventType (not all possible types are shown in the diagram), on which BPEExecutionElement (Activity, Sub-process, Gateway represented in execution), indicating the time of the occurrence, and its originator. Other information that can be registered is related to EventMeasures from registered EventLogs, such as processing time or cost of activities, etc.

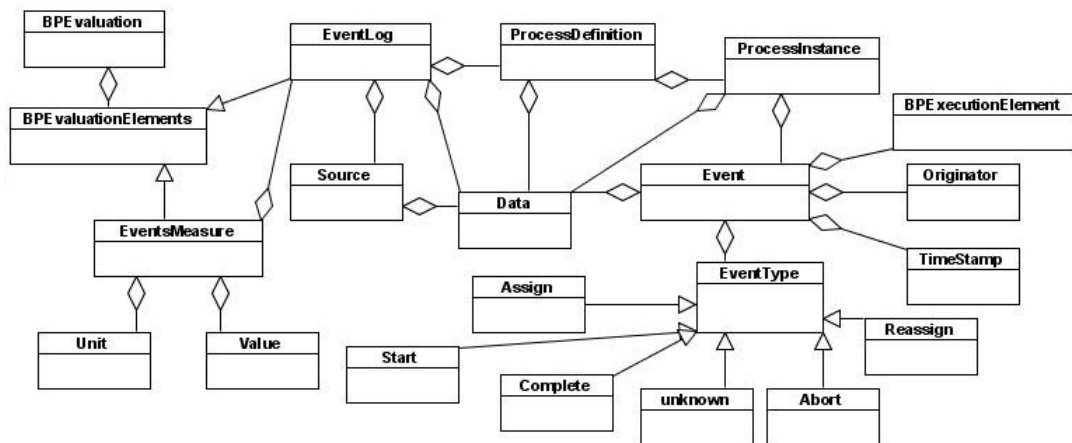


Figure 4.9.: BP Evaluation sub-Ontology (BPEVsO)

### Integration of the defined sub-Ontologies

The relevant relationships among the sub-Ontologies are presented, indicating the name of the relationship, the concepts it connects and its definition.

- **BPMsO and SOMsO relationships**

To illustrate the concepts and relationships that have been defined in diagram form, a UML class diagram of BPMsO and SMOsO is presented in Figure 4.10, and the relationships defined between their elements are described in Table 4.7. The relationships defined between the BP and service oriented modeling sub-ontologies, are of great importance in the context of our work, since they are the basis for the transformations from BP to services.

Figure 4.10 shows the elements defined to describe business processes at the top, those defined to describe services at the bottom, along with how the different elements are related to each other.

<sup>2</sup><http://www.processmining.org/logs/mxml>

<sup>3</sup><http://www.processmining.org/prom/start>

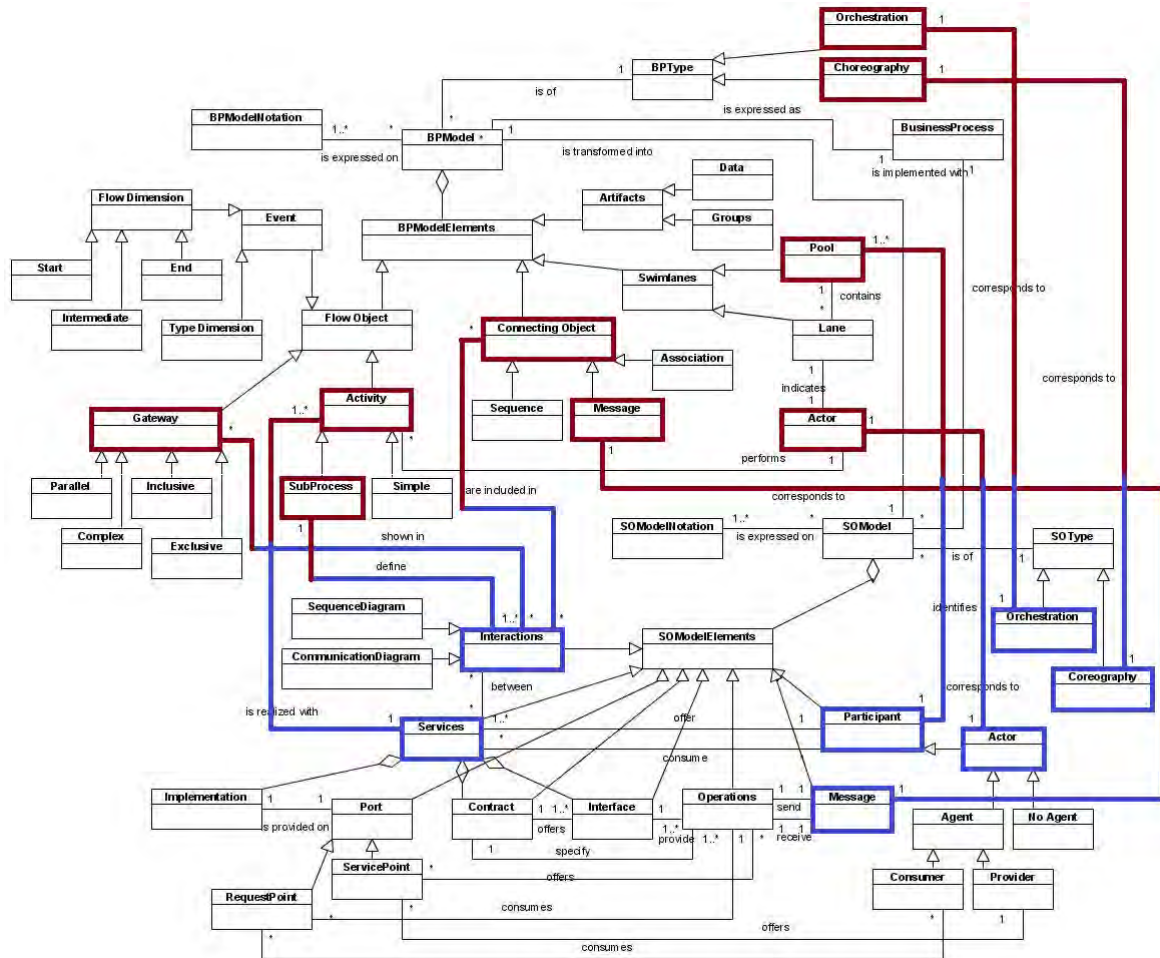


Figure 4.10.: UML class diagram for BPMsO and SOMsO integration

The relationships defined between the BPMsO and the SOMsO are described in Table 4.7, where, for example, it can be seen that there is a correspondence between a pool in a business process model and a participant in a service oriented model, a service corresponds to an activity (simple, sub-process), providing interfaces, operations, contracts, ports and an implementation. In BPMN2 these elements to model services were included so the correspondences changed to one-to-one relationships between these elements.

Other correspondences show how the flow of sub-processes can be represented by interaction diagrams including gateways and connecting objects, as well as how the messages needed for services to interact can be obtained from messages defined in the BP model, and how the type of services composition (orchestration, choreography) is determined by the type of BP model and their participants. Based on the concepts and relationships defined in the modeling sub-ontologies and the methodology for services development, we were able to define mappings between BPMN2 and SoaML metamodels.

An added value of the definition of the BPMsO and the SOMsO and their relationships is that of having the correspondences between elements from BPs and service models made explicit, which we have used as the basis for defining the transformations for the automatic generation of service models from BP models.

Table 4.7.: Summary of relationships defined between BPMsO and SOMsO

Name	Concepts	Definition
Corresponds-to	BPMsO.Pool- SOMsO.Participant	A pool in a business process model determines a participant in the service oriented model

Name	Concepts	Definition
Corresponds-to	BPMsO.Actor-SOMsO.Actor	An actor in a business process model determines different types of actors in the service oriented model to provide and consume services
Is-realized-with	BPMsO.Activity-SOMsO.Service	An activity in a business process model determines a service in the service oriented model, and depending on the type of the activity (sub-process, simple) it will be an atomic service or one composed by others
Defines	BPMsO.Sub-process-SOMsO.Interaction	A sub-process in a business process, besides of determining a service in the service oriented model also defines an interaction between the services that comprises it.
Shows-in	BPMsO.Gateway-SOMsO.Interaction	Gateways defining flows in a business process model, are shown in the flow of the services interaction that comprises the sub-process or process modeled.
Are-included-in	BPMsO.ConnectingObject-SOMsO.Interaction	The connecting objects in a business process are shown in the flow of the services interaction that comprises the sub-process or process modeled.
Corresponds-to	BPMsO.Message-SOMsO.Message	Messages in a business process determines in the service oriented model which are the needed messages for services to exchange for their realization
Corresponds-to	BPMsO.Orchestration-SOMsO.Orchestration	A business process model of type orchestration determines that the service oriented model will also be an orchestration of the associated services
Corresponds-to	BPMsO.Choreography-SOMsO.Choreography	A business process model of type choreography determines that the service oriented model will also be a choreography of the associated services between the identified participants

It is worth noting that in the BPMN v1.2 specification the notion of choreography was associated to the type collaboration process, this has changed in the BPMN2 standard since in this collaborative processes and choreographies are both defined as related but different views of the same concept.

- **SOMsO and SOEsO relationships**

The relationships established between the service oriented modeling and the service oriented execution sub-ontologies, represent mainly how the service design model is related to the technological implementation of defined services. In Table 4.8 the relationships defined for the SOMsO and the SOEsO are presented using the acronyms SOM for Service Oriented Model and SOE for Service Oriented Execution.

**Table 4.8.:** SOMsO and SOEsO relationships

Name	Concepts	Definition
Corresponds-to	SOMsO.ServicePoint-SOEsO.Provided	A ServicePoint port in SOM determines a Provided port in SOE.
Corresponds-to	SOMsO.RequestPoint-SOEsO.Requested	A RequestPoint port in SOM determines a Requested port in SOE.
Corresponds-to	SOMsO.Interface-SOEsO.Interface	A service Interface in SOMsO determines an interface in SOE which is used or implemented by a SOE Component.
Provided-by	SOMsO.Implementation-SOEsO.Interface	Implementation of a service in SOM is provided by one or more SOE Interfaces.
Implemented-as	SOMsO.Participant-SOEsO.Component	A Participant providing a service in SOM is implemented by a SOE Component

• **BPEVsO and BPEsO and BPMEsO relationships**

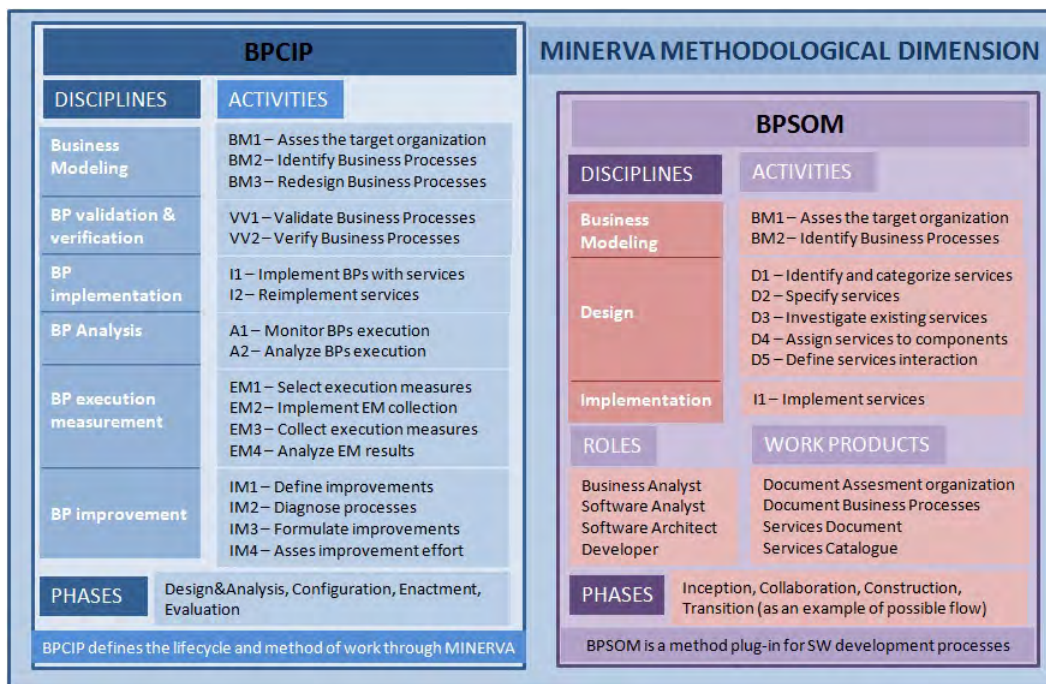
The relationships defined between the business process evaluation and business process execution and measuring sub-ontologies, represent on which elements the event logs are to be registered, as well as how measures grouping these event logs can be defined. In Table 4.9 the relationships defined for the BPEVsO with the BPEsO and BPMEsO are presented using the acronyms BPEV for Business Process Evaluation, BPE for Business Process Execution and BPME for Business Process Measuring.

**Table 4.9.:** BPEVsO and BPEsO and BPMEsO relationships

Name	Concepts	Definition
Corresponds-to	BPEsO.ExecutionElement- BPEVsO.ExecutionElement	The Event to register in BPEV is on an ExecutionElement that corresponds to the same in the BPE.
Corresponds-to	BPEsO.Process- BPEVsO.ProcessInstance	Each process execution in BPE corresponds to the ProcessInstance in BPEV.
Corresponds-to	BPEVsO.EventsMeasure- BPMEsO.Measure	Measures defined in BPEV grouping Events corresponds to BPME Measure.
Corresponds-to	BPEVsO.Unit- BPMEsO.UnitOfMeasurement	The unit to register a Measure in BPEV is defined as unit of Measure in BPME.
Corresponds-to	BPEVsO.Value- BPMEsO.MeasurementResult	The value of a measure in BPEV is a MeasurementResult of BPME.

**4.3.2. Methodological dimension**

In this dimension the methodological approaches are integrated into MINERVA as shown in Figure 4.11. BPCIP defines the elements to support the extended BP lifecycle with explicit measurement and improvement activities. BPSOM guides the development of service-oriented systems from BPs integrating the MDA approach defined to generate SoaML service models from BPMN2 models. In the following a brief description of each is given, just to introduce their main elements, as they will be presented in detail from chapter 5 to chapter 8.



**Figure 4.11.:** Methodological dimension elements

BPCIP defines the lifecycle and method of work through MINERVA to guide the management and improvement of BPs in the organization, providing a systematic way to identify, perform and integrate improvements both in BP models and/or the associated services implementation. It extends the BP lifecycle [Weske, 2007] with execution measurement activities and improvement activities we have defined based on the improvement process PmCompetisoft [Pino et al., 2009]. BPCIP also integrates a Business Process Execution Measurement Model (BPEMM) with a set of pre-defined execution measures to be used in the execution measurement activities within the phases defined. The BPCIP is presented in chapter 5 and the BPEMM in chapter 6.

BPSOM guides the service-oriented development from BPs, extending a previously defined methodology which established a core set of disciplines, activities, deliverables and roles to be added to the software development process used in the organization. It was also extended with the model-driven approach defined for the automatic generation of SoaML service models from BPMN2 models, as well as with the method and techniques needed for modeling BPs and services using these notations. BPSOM is presented in chapter 7 and the generation of SoaML service models from BPMN2 models in chapter 8.

### 4.3.3. Tools support dimension

The tool dimension gives support for working in all the phases of the MINERVA framework's defined lifecycle, manipulating all the elements defined in the Conceptual and Methodological dimensions, as shown in Figure 4.12. It integrates several existing tools along with newly developed ones, all of which are described in chapter 9.

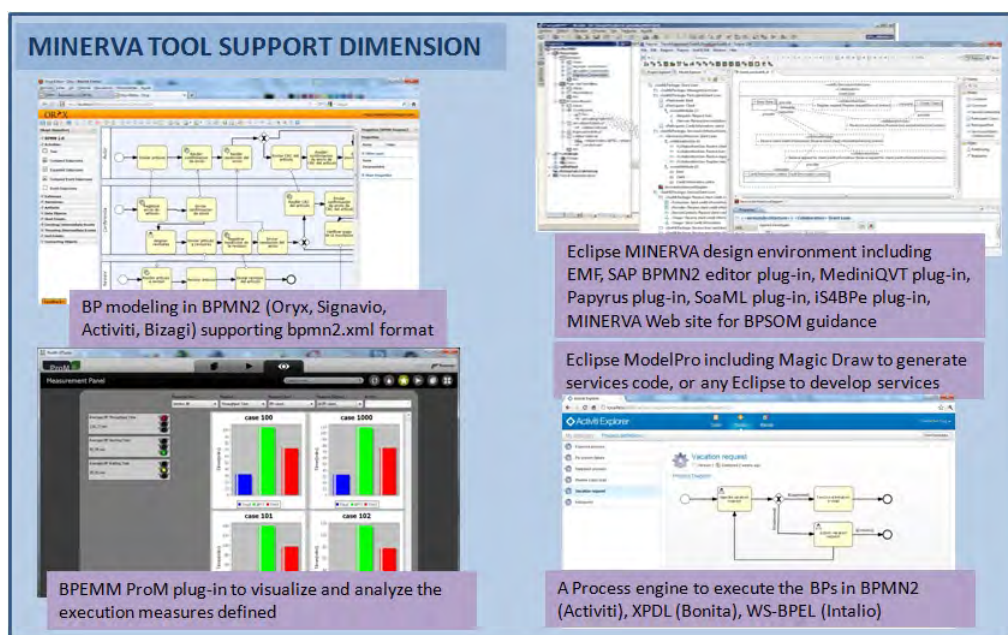


Figure 4.12.: Tool support dimension elements definition

For BP modeling we integrate the use of the Oryx<sup>4</sup> modeler, which is an academic open source project driven mainly by the Business Process Technology research group at the Hasso-Plattner-Institute, University of Potsdam, Germany. It provides complete support for BPMN2 modeling in a friendly web environment, and allows models to be saved in the BPMN2 exchange format. Other similar tools can be integrated as long as they enable models to be saved in some exchange format (BPMN2 or XPDL too).

For the generation and design of SoaML service models we provide the Eclipse MINERVA design distribution which integrates all the tools and plug-ins needed to perform work throughout the framework:

<sup>4</sup><http://bpt.hpi.uni-potsdam.de/Oryx/Research>

- from the existing Eclipse plug-ins, the EMF<sup>5</sup> for working with metamodels and models and validate them, the SAP<sup>6</sup> BPMN2 editor to manipulate BPMN2 metamodel and models and validating them, and the Eclipse MediniQVT<sup>7</sup> plug-in for defining and executing QVT transformations.
- from the newly developed Eclipse plug-ins, the Eclipse SoaML<sup>8</sup> plug-in to import the generated SoaML service model to visualize it or to design SoaML models from scratch (see chapter 9 for details), and the iS4BP<sup>e</sup> plug-in (insert Services for BP execution) which takes a SoaML generated service model and a BP input model (BPMN2, BPEL or XPD) and inserts the tags for invoking services in the corresponding activities (see chapter 9 for details).

For the execution of BPs in a process engine, one for each language was selected: Activiti<sup>9</sup> for BPMN2, Bonita<sup>10</sup> for XPD and Intalio<sup>11</sup> Community edition for WS-BPEL, based on a evaluation of process engines carried out among twelve selected ones (see chapter 9 for details and references).

Finally, to analyze BPs execution we have integrated the ProM<sup>12</sup> (Process Mining) framework, which is developed by the Process Mining Group, Math&CS department, Eindhoven University of Technology, The Netherlands. We have developed a ProM plug-in to present the results of the execution measurement results from BPEMM (see chapter 9 for details) in text and diagram form.

#### 4.4. Process view

The process view of the MINERVA framework defines the lifecycle of MINERVA framework by means of the Business Process Continuous Improvement Process (BPCIP) lifecycle, which is shown in Figure 4.13.

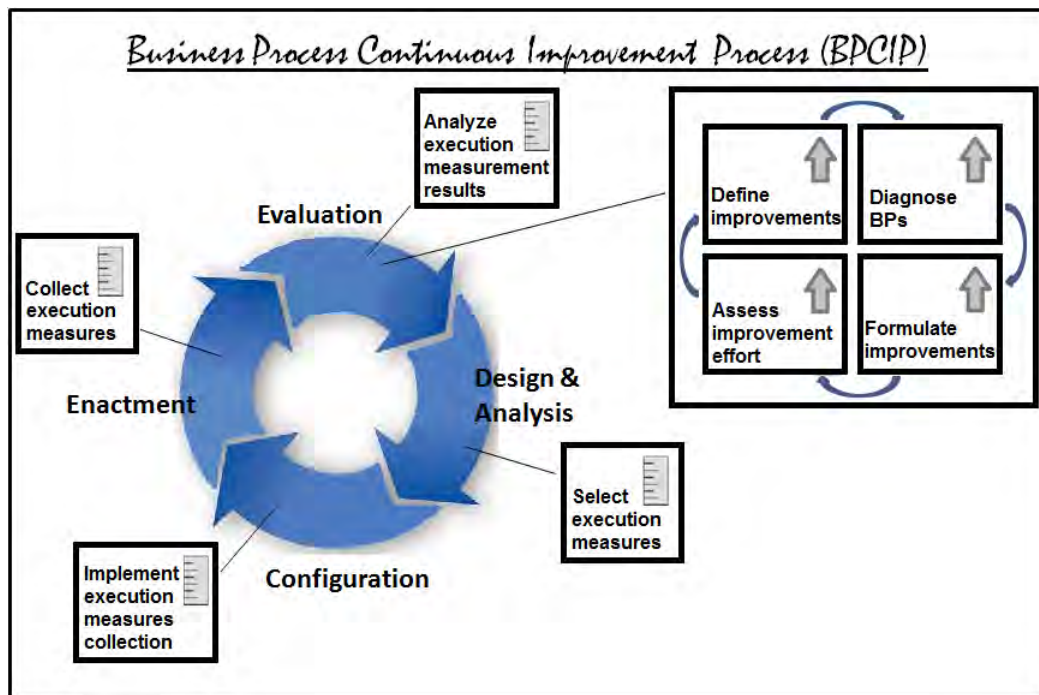


Figure 4.13.: MINERVA framework lifecycle

<sup>5</sup><http://eclipse.org/modeling/emf/>

<sup>6</sup><http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/c04f0691-0a76-2d10-1098-ec518f7bdf68>

<sup>7</sup><http://projects.ikv.de/qvt/>

<sup>8</sup><http://alarcos.esi.uclm.es/MINERVA/TOOLS/soamlPlugin.htm>

<sup>9</sup><http://www.activiti.org/>

<sup>10</sup><http://www.bonitasoft.com/>

<sup>11</sup><http://community.intalio.com/>

<sup>12</sup><http://www.processmining.org/prom/start>

In the first place we have added four explicit execution measurement activities (identified with an icon of a measurement rule in Figure 4.13) to the BP lifecycle [Weske, 2007], to provide measurement guidance over the different phases. These activities are performed as the final step in each BP lifecycle phase, to include the measurement vision over the BPs when they are modeled, implemented, deployed, executed and evaluated. This helps the stakeholders to know what they are meant to do in order to provide execution measurement support. The execution measurement defined activities are: Select execution measures, Implement execution measures collection, Collect execution measures and Analyze execution measurement results.

In the second place, we have also added four improvement activities (identified with an icon of an arrow pointing up in Figure 4.13) to guide the improvement effort in the organization, based on the results from the execution measurement of BPs and organizational guides provided in the Business Process Maturity Model (BPMM) [OMG, 2008b]. Once BP improvements opportunities are found in the Evaluation phase based on the analysis of the execution measurement results, the improvement activities we have defined are performed to guide the integration of improvements in the BP and services in a systematic way, which helps reach the business and improvement effort goals. The defined improvements activities are: Define improvements, Diagnose Processes, Formulate improvements and Asses improvement effort.

To assure the consistency of the work throughout the MINERVA framework we have defined a method of work through the BPCIP lifecycle described; an overview of this method is given in Figure 4.14. The red circled numbered path corresponds to the BP lifecycle plus the improvement activities we have integrated. The numbered path in purple squares shows the execution measurement activities we have added to the BP lifecycle, to put the focus on the measurement of BPs execution.

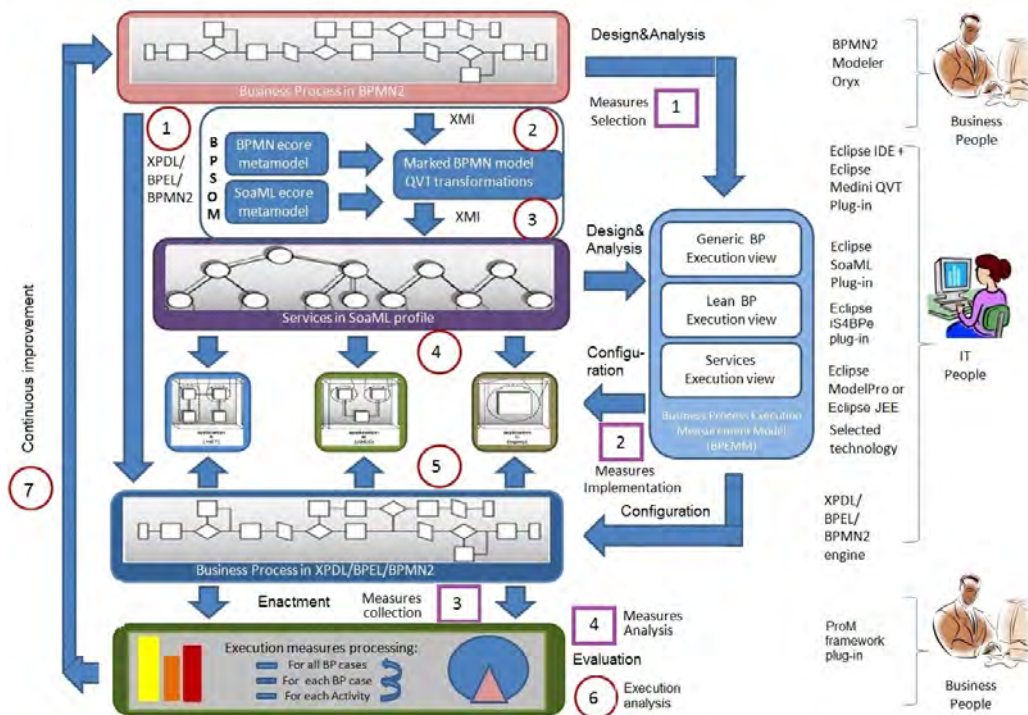


Figure 4.14.: MINERVA method of work through BPCIP

The method of work defines for each phase in the BPCIP lifecycle how the work is guided by MINERVA elements, including the measurement and improvement activities added. For each BP traversing the BPCIP lifecycle, we distinguish between the first iteration (iteration 0) where the first version of the BP is managed, and subsequent iterations (iteration  $n > 0$ ) where based on the improvements opportunities found, subsequent versions of the BP are managed. The MINERVA method of work is described briefly below, for each BPCIP phase and corresponding iterations.

- **Design & Analysis phase**

This phase aims to model and validate the BPs in the organization. In **iteration 0** the BP modeling (**top of figure**) is done by Business people and IT people (Analyst, Architect) in a BPMN2 modeler, applying best practices such as process patterns [van der Aalst et al., 2003a]. Business people also define specific goals for the BPs selecting execution measures from the BPEMM model, to be collected in the BPs execution (**purple square marked as 1**).

In **iteration n>0** when an improvement is being integrated, the BP is redesigned (**top of figure**) based on several existing proposals, to change the BP (and model) based on the information analyzed on the execution of the BP in the Evaluation phase, generating a new version of the BP to be executed. If there is no change in the BP model but rather on its implementation with services the redesign is skipped to continue to the Configuration phase.

- **Configuration phase**

This phase aims to develop, deploy and test the software to implement the BPs. In **iteration 0** the Software team (Analyst, Architect and Developers) carried out the service system development based on the BPSOM methodology and the model-driven approach, although other methodologies can be also used.

The BPMN2/XPDL/BPEL representation for the execution of the BP is obtained from the input BPMN2 model, by means of existing tools to transform one format into another (**red circle marked as 1**). The BPSOM methodology including the automatic generation of SoaML service models from the BPMN2 models is followed to develop the services needed (**red circle marked as 2 and 3**). After the services are generated we insert the invocation to them in the associated activities of the BPMN2/XPDL/BPEL for BP execution that we have generated before.

The SoaML service model generated is used to develop the corresponding code for executing the services in several technologies (JEE,WS) by means of existing model-driven engines or manually (**red circle marked as 4**). The previously selected execution measures are also implemented in the process engine and services implementation or infrastructure, to be able to collect the BP execution data needed (**purple square marked as 2**).

In **iteration n>0** when an improvement is being integrated, the selected service/s realizing the BP are reimplemented, based on the information analyzed on the execution of the BP in the Evaluation phase, or based on the changes performed in the previous phase generating the new version of the BP. If the BP model has been changed new services could be needed which are also developed to support its execution. The traceability between the BP model and the service model provided by means of the development guides and the automatic generation, allows the software team to know where the changes are to be made, as well as to gauge the associated impact.

- **Enactment phase**

This phase aims to execute the BPs registering data from the BP cases. In **iteration 0** the generated services are invoked from the process engine executing the BP (**red circle marked as 5**). The execution data needed to calculate the selected execution measures is collected by means of event log files (**purple square marked as 3**) from the BP engine and log functions implemented for services execution. In **iteration n>0** the same activities are carried out but for the new version of the BP generated in the previous phases.

- **Evaluation phase**

This phase aims to analyze and evaluate the execution of the BPs. In **iteration 0** the data in event logs is used to analyze BPs execution by means of the ProM framework (**red circle marked as 6**) and the BPEMM ProM plug-in we have developed (**purple square marked as 4**), to be used by business people to find improvement opportunities both in the BP and in the services execution. After analyzing the measurement results, a decision is made: when improvement opportunities are found the improvement activities we have



added are then performed to guide the integration of the improvement in the BP (**red circle marked as 7**), defining the improvements to be integrated to generate the new version of the BP, assessing the BP with BPMM and formulating the modifications to be made on the BP and/or service/s. After this another cycle through BPCIP is triggered to integrate the improvements and generate a new version of the BP.

In **iteration  $n > 0$**  after the improvements have been integrated in the BP and the new version of the BP has been executed and analyzed, the assessment of the improvement effort is carried out to compare the data from the execution of the previous version of the BP with the data from the execution of the new version of the BP, to assess if the goals of the integrated improvement have been achieved. In addition, the data registered about the improvement effort that has been carried out is analyzed, to identify improvement opportunities that may also exist in the improvement process that has been carried out.

## 4.5. Conclusions

In this Chapter the MINERVA framework has been presented which provides an integrated and standardized approach to support the management and continuous improvement of BPs realized by services with a model-driven approach. Its main elements have been described by means of the dimensions view defined: conceptual, methodological and tool support, which make up the structural view of the framework, as well as by means of the process view, which defines the BPCIP lifecycle and the method of work used throughout it.

Starting with the modeling of BPs in the organization, the MINERVA framework defines how to navigate from a BP model in BPMN2 to a service-oriented model in SoaML by means of both a methodological guide and the automatic generation of services through QVT transformations. From the generated service model the implementation of the services defined is obtained (automatically or manually based on the design definitions) which are in turn invoked from the BP execution in a suitable process engine. This execution is registered so to be able to calculate the execution measures defined to analyze the BP real behavior and to find improvement opportunities. These are integrated in a systematic way based on the improvement activities, thereby generating a new version of the BP by traversing the BPCIP cycle again.

MINERVA framework provides all the elements needed and a guide to support the complete BP lifecycle extended with measurement and improvement activities, aiming to integrate business and IT areas in a multidisciplinary effort that pursues the improvement of the way the organization performs its business, as well as of the technological and systems support for its execution.



## Chapter 5.

# Business Process Continuous Improvement Process (BPCIP)

This Chapter describes the Business Continuous Improvement Process (BPCIP) that has been defined in MINERVA framework to guide the improvement efforts for BPs implemented by services.

The Chapter is organized as follows: in section 5.1 a description of the BPCIP is presented, in section 5.2 the Disciplines and elements defined by BPCIP are set out and in section 5.3 the BPCIP lifecycle is described. section 5.4 describes the implementation of BPCIP as an EPF Composer method plug-in, and finally in section 5.5 conclusions for the chapter are discussed.

The contents of this chapter are complemented by the contents in the following chapters: chapter 6 which describes the Business Process Execution Measurement Model (BPEMM) defining BP and service execution measures to guide the measurement effort within the BPCIP process and chapter 7 which presents the Business Process Service Oriented Methodology (BPSOM) to guide the development of service-oriented systems from BPs with a model-driven approach.

### 5.1. Introduction

As presented in chapter 4 the Business Process Continuous Improvement Process (BPCIP) is an element of the methodological dimension of MINERVA, the main objective of which is to guide the execution measurement and improvement efforts in the organization, providing a systematic way to integrate improvements opportunities found in BPs and services implementation. It has been defined by extending the BP lifecycle [Weske, 2007] with explicit measurement activities and a BP Execution Measurement Model (BPEMM) -which is presented in chapter 6-, along with explicit improvement activities whose definition is based on the improvement process PmCompetisoft [Pino et al., 2009]. An initial definition of BPCIP [Delgado et al., 2011e] was updated based on feedback from the presentation to make the BPCIP lifecycle less complex than was proposed at first, which is the version presented here.

As it was mentioned in chapter 3 the BP lifecycle [Weske, 2007] defines the four phases of Design&Analysis, Configuration, Enactment and Evaluation of BP, outlining the main activities to be carried out. However, measurement and improvement activities are implicit in the BP lifecycle definition, being the collection of execution data in the form of a log file in the Enactment phase, the only one explicitly mentioned. It is our belief that it is necessary to define measurement and improvement activities explicitly, so as to be able to guide the execution measurement and improvement effort through the whole lifecycle. This will help obtain the insight needed about the real execution of BPs, and will assist in taking corresponding actions to improve them.

Based on this, the approach consists of traversing the BPCIP lifecycle to model, validate, simulate, implement, deploy, execute, analyze and evaluate BPs execution, including the selection, implementation, collection and calculation of the execution measures from BPEMM. When improvement opportunities are found based on the analysis of the data from BPs execution carried out in the Evaluation phase, the improvement activities are executed. These will define the improvements to be integrated, diagnose the BPs, and formulate the improvements, triggering a new execution of the BPCIP lifecycle to generate a new version of the BP. Finally, the results of executing the

previous BP version and the new one, respectively, are compared in order to evaluate the improvements; and the whole improvement effort is assessed to determine if the improvement goals have been achieved.

Disciplines and Phases have been defined in BPCIP in a similar way to how it is done in the Unified Process [Jacobson et al., 1999] to support the MINERVA method of work through the BPCIP lifecycle, as presented in chapter 4, guiding the continuous improvement effort based on BPs implemented by services driven by models. In Figure 5.1 the method of work of MINERVA as presented in chapter 4 is shown, to place the reader in the context of the definitions that are described below. The transparent boxes correspond to elements that although are mentioned here, will be described in next chapters.

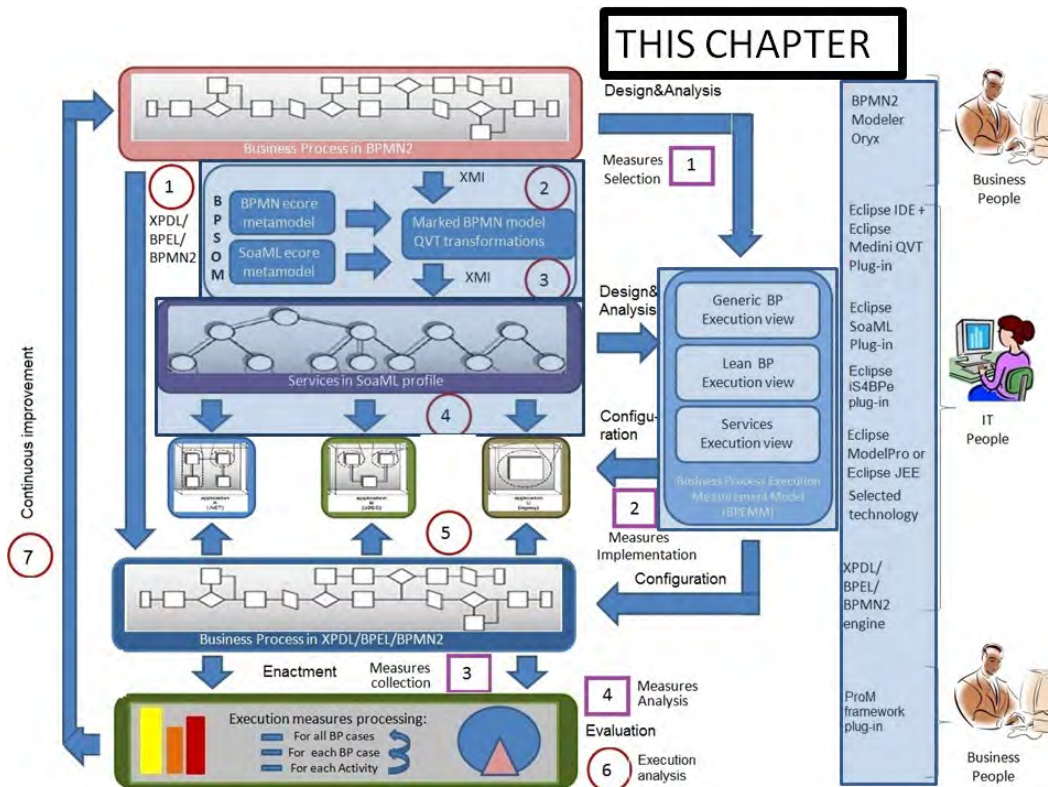


Figure 5.1.: MINERVA method of work through BPCIP

The BPCIP continuous improvement process has been also implemented<sup>1</sup> as an EPF Composer Eclipse [2004-2011] Method plug-in in order to provide interoperability with other processes defined in the same way, and published as a Web site [Delgado, 2011] to be easily accessed and used.

## 5.2. BPCIP Disciplines

The Disciplines of BPCIP are defined based on a “primary categorization mechanism for organizing tasks that define a mayor area of concern and/or cooperation of work effort” [RUP-IBM, 1999-2011], defining activities, artifacts and roles needed to guide the management and continuous improvement of BPs in the organization. The Disciplines and their activities, input and output artifacts and roles they comprise are presented below.

<sup>1</sup>Implementing a methodology means specifying it in a standard language in a way that means it could be enacted in an suitable process engine (SPEM engines)

### 5.2.1. Business Modeling Discipline

The Business Modeling Discipline aims to obtain a map of the organization and its BPs to gain a better understanding of the business by making their BPs explicit in models. To make the BPs of the organization in BP models explicit has several advantages: show how the BP are performed in the organization, what the specific internal participants of the BP (roles, sections) are, which activities are performed and how they are performed (manual, automatic), what the particular external participants of the BP (suppliers, clients, business partners) are and how they interact with the organization, which inputs and outputs are needed and generated by the BP, which resources are assigned and used, which events are managed, among others. BPs are also redesigned to integrate improvements after analyzing their execution or when validation and/or verification results are not as expected.

There are a great variety of notations for BP modeling Delgado et al. [2010g], although BPMN has, in recent years, emerged as the standard which is most preferred. This has occurred for several reasons, one being the fact that it is a notation which has its origins in the business area BPMI [2000-2005] and can thus be used and understood as common language by both the business and IT areas, helping close the gap between them. Based on these along with the fact that MINERVA is an standardized framework for construction, we have defined that the notation used to specify BP models is BPMN2.

In Figure 5.2 the activity detail diagram for the Business Modeling Discipline is shown presenting the main inputs and outputs and the role responsible for the activities, which corresponds to the Business Analyst, although the Responsible for the BP, as well as the IT Analyst and Architect also participate, as described below.

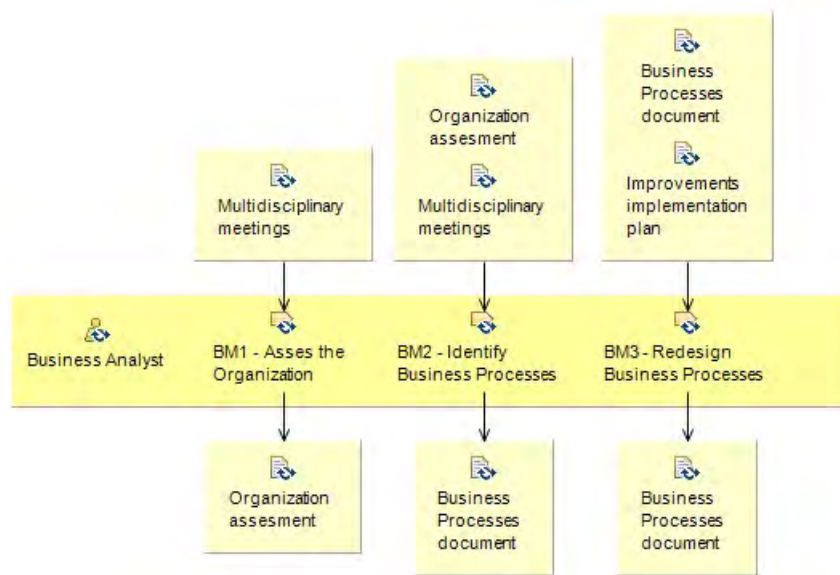


Figure 5.2.: Business Modeling activity detail diagram

#### 5.2.1.1. BM1 - Assess the Organization

This activity aims to provide insight into the business area of the organization, the business goals that have been set, and the business process that have been defined, together with how they operate. The objective is also to look at the employees and other participants involved (customers, competition), technologies used in the organization, problems and areas of improvements, among others, to make the main aspects of the BPs environment clear and explicit. The OMG Business Motivation Model (BMM) OMG [2010] can also be used to express the goals, strategy and the organization's other business information.

**Inputs** Information from the business area of the organization, mostly provided by the Business Analyst and the Responsible for the BP (for each BP discussed), along with technical information provided by the IT Analyst and Architect; during the meetings carried out.

**Outputs** Assessment of the Organization document, in which the information gathered is structured in sections to provide a clear view of the organization's current situation. This document will be discussed until all stakeholders agree upon it.

**Roles** The role responsible for this activity is the Business Analyst, and the Responsible for the BP; the IT Analysts and Architect also participate to provide their business and technical knowledge, respectively.

#### 5.2.1.2. BM2 - Identify Business Processes

The main objective of this activity is to identify and model BPs in the organization, including, among others, the activities carried out to perform the BP, how these activities are performed (manually, automatically), the control flow of the BP defined by the sequence of activities and the diverging and converging of flows based on decision nodes (gateways), the internal participants (roles, sections) and the external participants (suppliers, clients, business partners), inputs and outputs managed through the BP execution and the resources needed. BP models are specified in BPMN2

**Inputs** Assessment of the Organization document, including the information from the meetings between business and IT people about the current situation in the organization from the business and technical viewpoints.

**Outputs** The Business Process document in which BPs are identified and modeled including all the elements needed to specify these, along with the BPs in BPMN2 (XML) format.

**Roles** The role responsible for this activity is the Business Analyst, and the Responsible for the BP, and IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

#### 5.2.1.3. BM3 - Redesign Business Processes

This activity is executed mainly in the context of an improvement effort after improvement opportunities have been detected for the BP based on the evaluation of the BP real execution from an analysis of that execution. It can also be performed after a validation and/or verification of the BP model has been carried out, in order to include improvements found prior to its implementation. In this case, the validation and/or verification documents are also an input for the activity, which are not shown in Figure 5.2 as they are marked as optional inputs.

The redesign of the BP model needs to take into account several proposals and approaches [Reijers, 2003, Maruster and van Beest, 2009, Netjes, 2010], as well as the tradeoffs between different aspects (such as the “devil’s quadrant” dimensions that are explained in chapter 6). This is done to assure that the modification introduced to fulfill the improvement goals is sufficient. The modifications can affect the BP model and/or also the real executed BP, to align the real BP with its model.

**Inputs** The Business Process document in which the BPs are identified and modeled, along with the BP models in BPMN2 format, as well as the modifications defined for the BP by means of the improvements that are to be integrated in the BP in the improvement plan. If the redesign is performed after a validation and/or verification of the BP model has been carried out, the validation and/or verification document are also inputs.

**Outputs** The Business Process document in which the new version of the BP is registered, along with the new version of the BP in BPMN2 (XML) format.

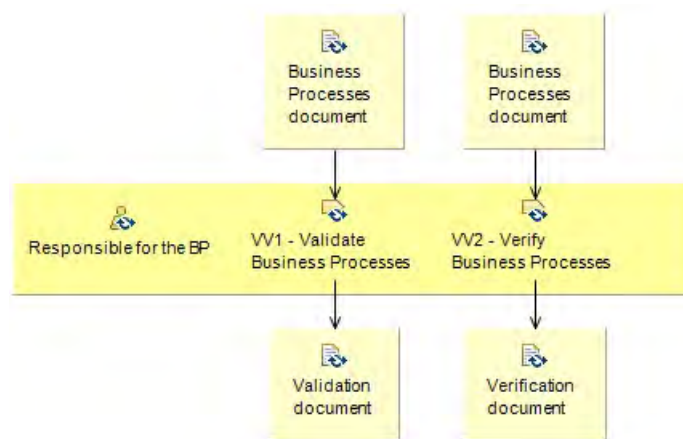
**Roles** The role responsible for this activity is the Business Analyst; the BP responsible and the IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

### 5.2.2. Business Process Validation & Verification Discipline

The Business Process Validation Discipline aims to validate the BP model prior to its implementation, to be able to detect improvement opportunities earlier in the BP lifecycle. Several validation techniques can be used to validate the model and these are mainly classified into two distinct groups: analytical techniques and simulation. The first group includes mathematical theories such as queuing models and stochastic models while the second group involves experimentation and is computer based [Laguna and Marklund, 2005, van der Aalst and Voorhoeve, 2010-2011]. They allow to answer questions of the “what-if” type about the BP execution, by analyzing different possible scenarios of the BP execution.

Another type of validation includes, for example, the evaluation of several quality characteristics of the BP model, such as complexity, coupling or cohesion. Various design measures that can be integrated into the validation effort have been defined to assess these characteristics [Cardoso et al., 2002, Rolón et al., 2006, Mendling, 2008, Sánchez González et al., 2010]. On the other hand, the verification of properties in the models could anticipate unwanted situations such as the existence of deadlocks, or determine the soundness of a BP model. We leave it up to the BPCIP user (organization) to perform validation and verification on the BP model, as well as to choose the validation and verification techniques to be used, since it depends on what each organization is willing and able to perform.

In Figure 5.3 the activity detail diagram for the BP Validation and Verification Discipline is shown, with the responsible role defined, corresponding to the Responsible for the BP, although the Business Analyst and the IT Analyst and Architect are also involved in the activities defined, as presented below.



**Figure 5.3.:** BP validation and verification activity detail diagram

#### 5.2.2.1. VV1 - Validate Business Processes

The validation approach is selected, whether simulation or analytical techniques, to gain insight into the characteristics of the BP model prior to its implementation. The use of analytical techniques needs a high degree of expertise in the techniques to carry out the validation; simulation, for its part, is based on tools which provide the environment needed to run the simulation of the

**BPs.** The definition of the different scenarios to be simulated also needs expertise but from the business and the specific BP (i.e. to define values for the simulation: rate of arrival of new cases, duration of activities, use of resources, probabilities of decision flows, among others). Several quality characteristics such as complexity, coupling or cohesion which provide extra information about the BP model, can be also assessed in the BP model.

**Inputs** The Business Process document in which the business processes are identified and modeled, along with the BP models in BPMN2 format.

**Outputs** The Validation document that includes the validation information (for example in the case of simulation this information refers to the configuration of the parameters for each simulated scenario) along with the results of the validation performed.

**Roles** The role responsible for this activity is the Responsible for the BP, but the Business Analysts, the IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

#### 5.2.2.2. VV2 - Verify Business Processes

The verification of BPs models is performed in order to gain insight into several characteristics of the model prior to its implementation, detecting deadlocks or determining its soundness, for example. Several techniques and tools can be selected to perform this kind of verification. It may be necessary to transform the BPMN2 model into a suitable notation, such as Petri nets, to carry out this verification of characteristics.

**Inputs** The Business Process document in which the business processes are identified and modeled, along with the BP models in BPMN2 format.

**Outputs** The Verification document that includes the verification information (for example which characteristics will be verified and in what way) and the results of the verification performed.

**Roles** The role responsible for this activity is the Responsible for the BP, but Business Analysts, the IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

#### 5.2.3. Business Process Implementation Discipline

The Business Process Implementation Discipline aims to implement BPs with services driven by models by means of the BPSOM methodology, although other methodologies can also be used. This implementation implies to obtain service models from the BP model to be able to develop the services to realize the BP. Services are also reimplemented to integrate improvements based on the analysis of the execution of the BPs.

We provide the BPSOM methodology to guide the development of SoaML service models from BPMN2 models (with automatic generation or manually), which is described in chapter 7, although other methodologies or approaches can be used, as long as they provide the implementation of services related to BPs to be invoked from the executable BP. We also provide tool support for performing the activities in BPSOM including the Eclipse SoaML plug-in we have developed for SoaML modeling, described in chapter 9.

It also implies making the BP model executable, which depending on the language used by the selected process engine (BPMN2/XPDL/BPEL) needs further work to be done on the BP model.



Several existing tools can be used to generate XPLD/BPEL models from BPMN2 models that can be integrated to perform this transformation. Nevertheless we provide support for the addition of the invocation to the services generated to the executable BPMN2/XPDL/BPEL model, by means of the Eclipse iS4BP (insert services for BP execution) plug-in we have developed, described in chapter 9.

In Figure 5.4 the activity detail diagram for the BP Implementation Discipline is shown, with the responsible role defined, corresponding to the Architect, although other roles such as the Analyst and the Developer, are involved in the activities defined, as presented below.

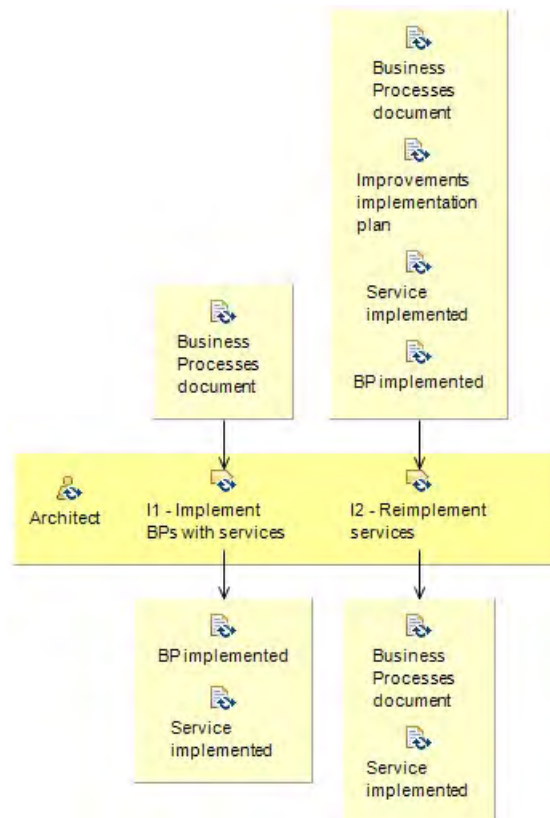


Figure 5.4.: BP implementation activity detail diagram

### 5.2.3.1. I1 - Implement BPs with services

This activity defines the carrying out of two main tasks: first of all, there is the generation of the executable BPMN2/XPDL/BPEL from the BPMN2 model specified in the Business Modeling Discipline, and secondly, the development of service models from the same BPMN2 model making it possible to implement the services to realize the BP. For the development of services from BPMN2 models we provide the BPSOM methodology (see chapter 7) which also produces other intermediate artifacts such as the SoaML service models, the Services document and the Services catalogue to centralize the definition of services throughout the organization.

For the implementation of the BP model for execution, if the selected language is BPMN2, then all the elements in the BP model have to be defined and implemented, such as, among others, user forms, invoking the implemented services, include scripts or business rules calculation by a rules engine, defining the conditions in the gateways. If the language selected is XPDL or WS-BPEL, first a transformation from the BPMN2 model to obtain the corresponding XPDL/BPEL model has to be performed, and then the model obtained has to be made executable by defining all the elements involved, in the same way as for BPMN2.

**Inputs** The Business Process document in which the BPs are identified and modeled, along with the BP models in BPMN2 format.

**Outputs** The BP implemented in the language defined by the process engine to execute the BP model (BPMN2/XPDL/BPEL), including the invocations to the services implemented to realize the BP, and the services implemented. When using BPSOM to guide the service-oriented development, other artifacts are also generated such as SoaML service models, the Services document and the Services catalogue.

**Roles** The role responsible for this activity is the Architect, but the Analysts and Developer roles also participate, providing their business and technical knowledge, respectively.

#### 5.2.3.2. I2 - Reimplement services

This activity is carried out in the context of an improvement effort, when an improvement opportunity has been found for the implementation of the services defined. The corresponding specification of services has to be updated accordingly, taking into account the definitions of the BP as specified in the BP document. The definitions in the Improvement implementation plan constitute the guide for the modifications to be carried out in the service/s.

**Inputs** The Business Process document in which the business processes are identified and modeled, along with the BP models in BPMN2 format, the implementation of the service/s and the Improvements implementation plan in which the improvements to be integrated into the service/s are specified.

**Outputs** The service/s implemented including the integrated improvements and the corresponding update of the service/s specification and its relationship with the corresponding BP model. If something in the BP model has also been modified then the updated BP document is another output of this activity.

**Roles** The role responsible for this activity is the Architect, but the Analysts and Developer roles also participate, providing their business and technical knowledge, respectively.

#### 5.2.4. Business Process Analysis Discipline

The BP Analysis Discipline aims to analyze the execution of BPs both in real time by means of monitoring BPs execution, as well as after a considerable amount of BP instances have been executed, registering the corresponding data. Based on the data provided in real time immediate decisions can be taken by business people to improve the execution of BPs at that time, such as assigning more resources if a bottleneck is detected. It also provides information about the status of the BP that can be useful for the Client or partners of the BP.

The second kind of analysis aims to process data from the execution of the BPs to provide the business area with information to enable it to find improvement opportunities for the BP, whose integration will result in the generation of a new version of the BP to be deployed in the organization. Several Business Intelligence (BI) tools and approaches such as Process Mining (PM) techniques provided by the ProM framework can be used, and we leave the corresponding selection up to the BPCIP user (organization). The ProM framework and its several plug-ins is recommended in this activity, however, we also provide a ProM plug-in to analyze BPEMM execution measures, that is used in the BP Execution Measurement Discipline, which is described in the next subsection.

In Figure 5.5 the activity detail diagram for the BP Analysis Discipline is shown, with the responsible role defined, corresponding to the Responsible for the BP, although other roles are involved in the defined activities, as presented below.

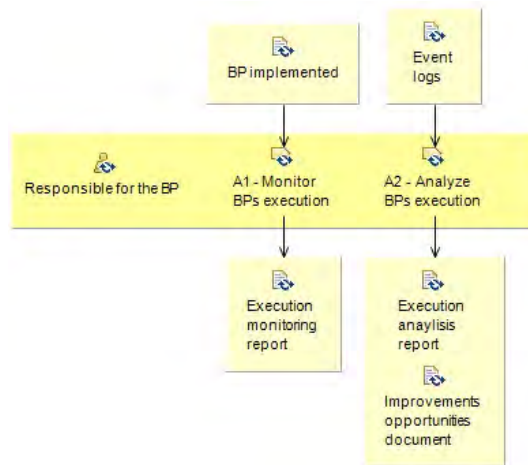


Figure 5.5.: BP Analysis activity detail diagram

#### 5.2.4.1. A1 - Monitor BPs execution

The monitoring of BPs execution is performed on the basis of the software products which provides real time access to BPs execution data, such as the current state of the BP case and its activities, as well as to key performance indicators that allow the organization to take immediate actions to improve the performance of the BP.

Monitoring BPs execution helps the organization to carry out, for example, reassignment of resources to BPs along with detection of malfunctions in the software supporting the execution, thus providing another view into the long term improvements that can be integrated into the BPs.

**Inputs** The BP implemented as it executes in the process engine in accordance with the BP model.

**Outputs** A BP monitoring report that is delivered by the BAM software by means of the screens showing the real time information on the BP execution, along with facilities to export the information into a suitable format for documentation (i.e. excel).

**Roles** The role responsible for this activity and the only one performing it is the Responsible for the BP, which is in charge of its operation and the monitoring of its execution.

#### 5.2.4.2. A2 - Analyze BPs execution

From the data registered in the event logs and services/systems logs for each BP case executed, several Process Mining (PM) techniques provided by the ProM framework can be used to analyze the execution of the BP based on three different perspectives: discovering BP models from the BP execution, checking the conformance of existing BP models against the BP real execution, and extending or improving existing BP models with other information from the BP execution, such as resources.

The analysis carried out by means of the ProM plug-ins for Process Mining (PM) and the ProM BPEMM plug-in for the calculation and visualization of the BP execution measurement results (performed in the Execution Measurement Discipline), is used as a basis for business people to be able to find improvement opportunities for the BP.

**Inputs** The event log corresponding to the BP execution in the process engine.

**Outputs** An execution analysis report in which the results for the application of different PM perspectives by means of the existing ProM plug-ins are described (if performed).

**Roles** The role responsible for this activity is the Responsible for the BP who is in charge for its operation, the Business Analyst and the IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

### 5.2.5. Business Process Execution Measurement Discipline

The BP execution measurement Discipline is defined to show explicitly the execution measurement activities to be performed extending the BP lifecycle, in the context of the BPCIP. We have defined this discipline following the vision proposed in the Capability Maturity Model Integration (CMMI) [ref] in which an explicit Measurement Process Area is defined with specific activities to guide the measurement effort.

The activities defined provide the basis for guiding the execution measurement effort around the BP modeling, implementation, execution and evaluation, by means of specifying the related measurement tasks to be performed, including the artifacts and roles involved. Four activities are defined to extend the BP lifecycle with explicit execution measurement.

In Figure 5.6 the activity detail diagram for the BP Execution Measurement Discipline is shown, with the role responsible defined, corresponding to the Responsible for the BP dealing with the measurement activities related to the business area, and to the Developer for the activities related to the technologies, although other roles are involved in the activities defined, as presented below.

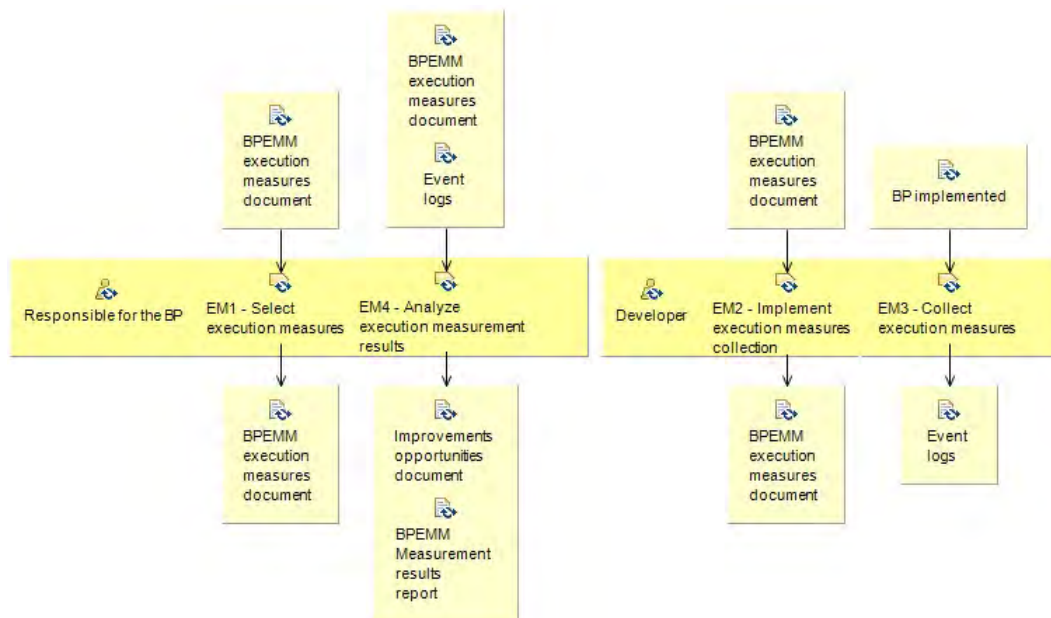


Figure 5.6.: BP Execution Measurement activity detail diagram

#### 5.2.5.1. EM1 - Select execution measures

This activity is carried out to define which execution measures will be calculated from the execution of the BP, selecting them from the ones integrated in the BP Emm model. The base measures defined for each goal in the BP Emm provides the data that has to be collected from the BP execution to calculate the rest of the measures defined.

**Inputs** The BPEMM execution measures document which provides the set of execution measures for each goal defined for the organization and specific to the BPs.

**Outputs** The BPEMM execution measures document providing the information about the execution measures selected for each BP.

**Roles** The role responsible for this activity is the Responsible for the BP which is in charge for its operation, the Business Analyst and the IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

#### 5.2.5.2. EM2 - Implement execution measures collection

In this activity the execution measures from BPEMM have to be implemented in the infrastructure in which the BP and the services will be executed. This means integrating the base measures that defines the data to be collected into the process engine and the infrastructure in which the services will execute.

**Inputs** The BPEMM execution measures document which provides the information on the base measures to be implemented and the execution measures selected for each BP.

**Outputs** The BPEMM execution measures document providing the information on the implementation of the execution measures.

**Roles** The role responsible for this activity is the Developer who is in charge of implementing the execution measures in the process engine and infrastructure for executing services, the Architect also participates to provide his/her technical knowledge.

#### 5.2.5.3. EM3 - Collect execution measures

The data for the execution measures selected from BPEMM is collected in the process engine and the infrastructure executing services, as the BP cases execute, registering data such as the start and completion time of activities, as well as the corresponding performers, and the data involved, to be able to calculate the execution measures from BPEMM.

**Inputs** The BP implemented and executing in a process engine invoking the associated services in the infrastructure defined.

**Outputs** The event log corresponding to the execution of the BP and associated services.

**Roles** The role responsible for this activity is the Developer who is in charge of implementing the execution measures in the process engine and infrastructure for executing services, the Architect also participates to provide his/her technical knowledge.

#### 5.2.5.4. EM4 - Analyze execution measurement results

To analyze the execution measurement results as defined in BPEMM, the execution measures are calculated and visualized by means of the ProM plug-in we have developed, providing information on the three views defined, which are described in chapter 6. It includes graphics and numeric results, allowing to drill up and down within the hierarchy defined, to analyze the results. Based on the measurement results for the BP the business area can find improvement opportunities to be integrated into the BP to achieve the business and BP goals defined.

**Inputs** The execution event logs from the BP execution and the BPEMM measurement results report generated in the form of the ProM plug-in showing the results for the BP, and the BPEMM execution measures document where the formulae for the measures are specified.

**Outputs** The Improvements opportunities document in which the improvements found for the BP are specified.

**Roles** The role responsible for this activity is the Responsible for the BP which is in charge for its operation, the Business Analyst and the IT Analyst and Architect also participate, providing their business and technical knowledge, respectively.

### 5.2.6. Business Process Improvement Discipline

The BP Improvement Discipline is defined to show explicitly the improvement activities to be performed in the context of the BPCIP. The activities defined provide the basis for guiding the improvement effort for integrating improvement opportunities into the BPs, in a systematic way. Four activities are defined to be performed after improvement opportunities for the BP have been found, with the focus being on an agile integration minimizing the effort of carrying out the improvement process, in order to make it easier to use. The definition of the improvement activities is based on the improvement process PmCompetisoft [Pino et al., 2009] which was defined taking into account these objectives with a focus on VSEs.

In Figure 5.7 the activity detail diagram for the BP Improvement Discipline is shown, with the role responsible defined, corresponding to the Responsible for the Improvement, although other roles are involved in the defined activities, as presented below.

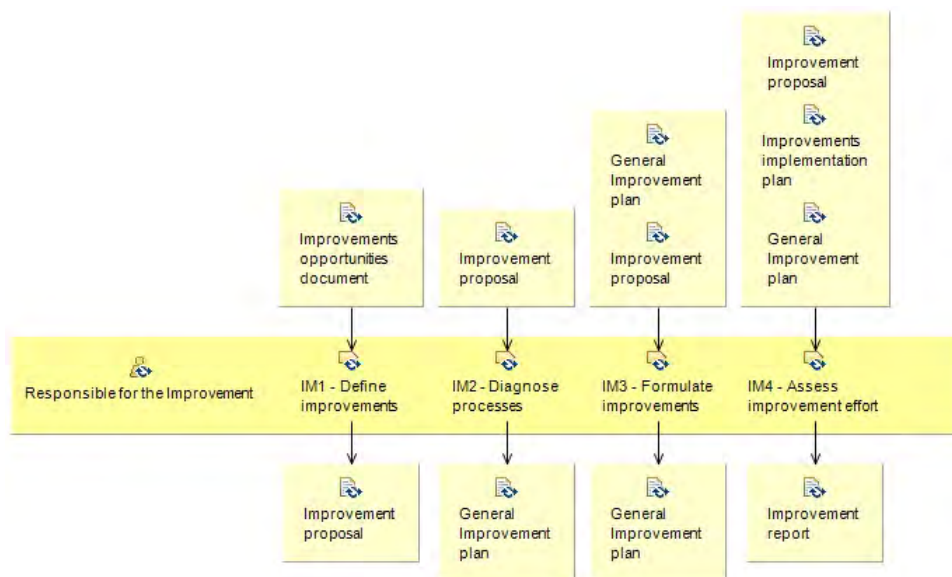


Figure 5.7.: BP Improvement Discipline activity detail diagram

#### 5.2.6.1. IM1 - Define improvements

This activity implies stating which improvements will be carried out to integrate the improvement opportunities found by analyzing the execution of the BP. The improvements to be integrated in the BP are defined explicitly, along with the objectives for the improvement effort and information about the needed resources for carrying it out. The improvement proposal is aligned with the strategic business goals of the organization and the specific goals for the BP, and has to be approved by the management area to ensure the sponsorship for the improvement.

**Inputs** The Improvements opportunities document which specifies the improvements found for the BP based on the analysis of the real BP execution.

**Outputs** The Improvement proposal which defines the improvements to be integrated in the BP and the specific information on the improvement effort to be undertaken.

**Roles** The role responsible for this activity is the Responsible for the Improvement which is in charge of the improvement effort, the Responsible for the BP which is in charge for its operation, the Improvement Group and the Business Analyst also participate.

#### 5.2.6.2. IM2 - Diagnose Processes

The diagnosis of the BP using the BPMM standard implies reviewing the definition of the BP and the way it is being carried out in the organization, to get insight into its maturity level. The maturity of the BP is evaluated against the BPMM definitions by performing an internal evaluation, aiming to gain insight into organizational aspects of the BP definition such as management.

As described in chapter 3, the BPMM follows the format defined by the software maturity models (CMM, CMMI) and includes several Process Areas and defined Key Activities, that when performed, allow the BP to gain maturity by evolving through the model's five maturity levels.

**Inputs** The Improvement proposal which defines the improvements to be integrated in the BP, as well as the specific information on the improvement effort to be undertaken.

**Outputs** The General improvement plan which includes the results of the BPMM assessment carried out for the BP, as well as the new improvement opportunities found.

**Roles** The role responsible for this activity is the Responsible for the Improvement which is in charge of the improvement effort, the Responsible for the BP which is in charge for its operation, the Improvement Group and the Business Analyst also participate.

#### 5.2.6.3. IM3 - Formulate improvements

This activity aims to define explicitly which part or parts of the BP model will be modified, as well as to identify explicitly the part or parts of the service or services realizing the BP, to achieve the improvements defined. To do so, the changes have to be defined specifically, i.e., if the execution time of an activity in a BP has been identified as being longer than it should be and its execution time should be improved, it could be specified that for this activity several redesigns must be evaluated to obtain better results. The same applies if the problem detected involves the execution of services which realizes the BP.

**Inputs** The General improvement plan which specifies the prioritized improvements that are to be integrated into the BP.

**Outputs** The Improvement implementation plan which explicitly specifies the part/s of the BP model and/or services in which the improvements are to be integrated and how.

**Roles** The role responsible for this activity is the Responsible for the Improvement which is in charge of the improvement effort, the Responsible for the BP which is in charge for its operation, the Business Analyst, the Improvement Group, the IT Analyst and Architect also participate.

#### 5.2.6.4. IM4 - Assess improvement effort

This activity implies evaluating the achievement of the goals defined for the improvement effort, in terms of the improvement of the BP as defined, as well as regarding the schedule, resources and cost defined for the cycle. It also includes the realization of a post-mortem analysis to assess the development of the improvement cycle, to ensure that all the information is registered and to add information such as lessons learned and improvements opportunities for the improvement cycle itself.

**Inputs** The improvement documentation generated through the improvement cycle comprising: the Improvement proposal, the General improvement plan and Improvement implementation plan. The report on the comparison of execution versions, which includes the comparison between the execution of the new BP version and the previous one.

**Outputs** The Improvement report which includes all the information for the post-mortem analysis performed for the improvement effort.

**Roles** The role responsible for this activity is the Responsible for the Improvement which is in charge of the improvement effort, the Responsible for the BP which is in charge for its operation, the Improvement Group and the Business Analyst also participate.

### 5.3. BPCIP Phases

BPCIP phases define the lifecycle of MINERVA, which begins with the modeling of a new BP or with redesigning an existing one in BPMN2, implementing it by means of services based on models, whose execution is then measured and evaluated based on BPEMM execution measures, aiming to identify improvement opportunities. These improvements can then be fed back into the BP following a systematic approach based on the improvement activities defined.

As presented in the previous section, the four explicit execution measurement activities added to the BP lifecycle provide measurement guidance and support over the different phases. These activities are performed as the final step in each BP lifecycle phase, to include the measurement vision over the BPs when they are modeled, implemented, deployed, executed and evaluated. The four explicit improvement activities added to the BP lifecycle are performed in the Evaluation phase after the improvement opportunities have been found, to guide their integration in a systematic way generating a new version of the BP by traversing the BPCIP lifecycle once more.

Finally, the measures of the new version of the BP with the integrated improvements can be compared with the one from the previous version, to evaluate the results of the changes carried out. The lifecycle provides the guide to indicate the emphasis on the realization of activities at each stage of the management and improvement of BPs in the organization. In Table 5.1 a summary of the activities related to the Disciplines and Phases is shown, following the UP style.

**Table 5.1.:** Summary of activities in Disciplines and Phases

Disciplines	Phases			
	Design&Analysis	Configuration	Enactment	Evaluation
Business Modeling	BM1-Assess the organization BM2-Identify BPs BM3-Redesign BPs			
BP Validation &Verification	VV1-Validate BPs VV2-Verificate BPs			



Disciplines	Phases			
	Design&Analysis	Configuration	Enactment	Evaluation
BP Implementation		I1-Implement BPs with services I2-Re-implement services		
BP Analysis			A1-Monitor BP execution	A2-Analyze BP execution
BP execution measurement	EM1-Select execution measures	EM2-Implement execution measures	EM3-Collect execution measures	EM4-Analyze execution measurement results
BP improvement				IM1-Define improvements IM2-Diagnose processes IM3-Formulate improvements IM4-Assess improvement efforts

### 5.3.1. Design&Analysis phase

The Design&Analysis phase puts the emphasis on the definition of the organizational environment including the business and technical context, as well as the design and specification of BPs by means of BPMN2. For each BP, the first iteration aims to obtain the corresponding BPMN2 model, and subsequent iterations aim to redesign it integrating improvement opportunities found in the Evaluation phase.

In both cases, models can be validated through simulation or analytical techniques to assess if they allow the business goals defined for the BP to be achieved, or to help evaluate different design options for it. Moreover, design measures can be used to assess quality characteristics of the model created (i.e., complexity) as well as to detect potential problems in early stages.

Finally, the BPEMM is used to select execution measures according both to the business objectives defined for the BP and the business strategy of the organization. Figure 5.8 shows the activity diagram for the Design&Analysis phase.

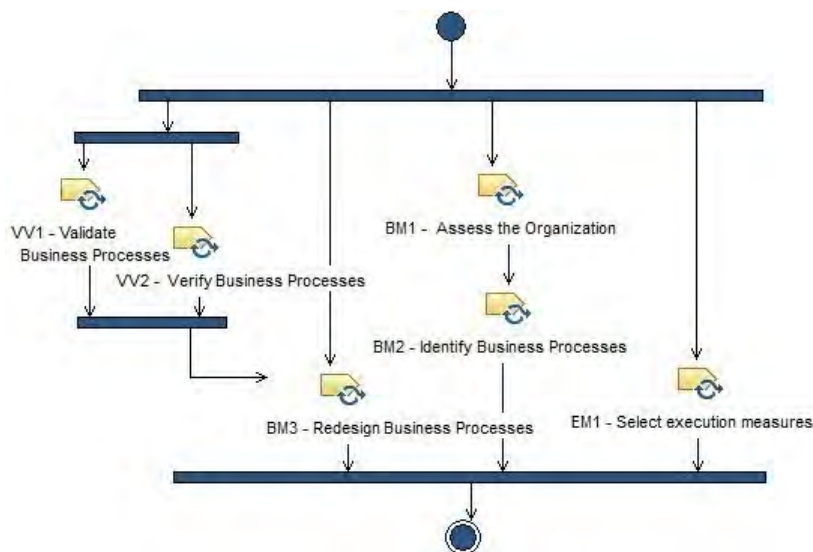


Figure 5.8.: BPCIP Design&Analysis phase activity diagram

### 5.3.2. Configuration phase

In the Configuration phase the BP models are made executable in a suitable language (BPMN2/ XPD/ BPEL) to be executed in the selected process engine, and are implemented by services. For each BP the first iteration aims to obtain an executable BP model from which to invoke the services implemented; subsequent iterations aim to re-implement the service/s so as to integrate improvement opportunities found in the Evaluation phase. To guide the service-oriented development from BPMN2 models generating SoaML service models we provide the BPSOM methodology, although other methodologies can be used.

This phase is also concerned with the implementation of the execution measures selected from BPEMM to be integrated directly into the process engine and into the services/software systems, to register the data needed to calculate the execution measures. Figure 5.9 shows the activity diagram for the Configuration phase.



Figure 5.9.: BPCIP Configuration phase activity diagram

### 5.3.3. Enactment phase

In the Enactment phase the BPs are executed in the selected process engine in the corresponding language (BPEL/ XPD/ BPMN2), invoking the services realizing the BP. The BPEMM execution measures defined and implemented are collected during the execution of BP cases (instances), registering the data needed for the execution measures to be calculated later. The data is put together in execution event logs which provide the basis for performing BPs execution analysis in the next phase of Evaluation. The execution of the BPs is monitored in real time to provide immediate information about their execution, allowing business people to make improvement decisions as BP cases are executed. Figure 5.10 shows the activity diagram for the Enactment phase.

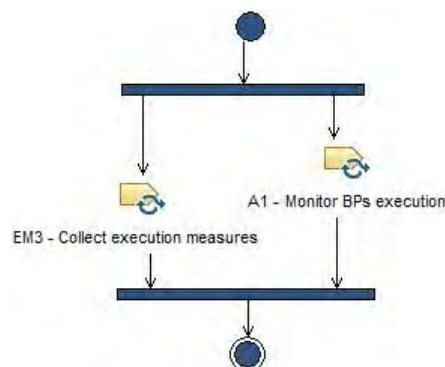


Figure 5.10.: BPCIP Enactment phase activity diagram

### 5.3.4. Evaluation phase

BPs execution is analyzed in the Evaluation phase by means of the execution data registered in the event logs. The execution measures from BPEMM are calculated on the basis of the data registered in the execution logs by means of the ProM framework plug-in we have developed. Using several other existing ProM plug-ins, different views of the associated data can also be analyzed. Using the analysis performed it is possible to identify improvement opportunities for the BP, which can be related to the BP modelling level as well as to the services realizing the BP, such as bottlenecks in the BP or service execution delays.

When improvement opportunities are found the improvement activities we have added are performed, to guide the integration of improvements in a systematic way, generating a new version of the BP and triggering a new execution of the BPCIP lifecycle. Improvements are integrated and the new version of the BP is deployed, executed and analyzed, then compared with the execution of the previous version, to see if the improvement goals have been achieved. The execution of the improvement effort is also evaluated to find improvements in the process itself. Figure 5.11 shows the activity diagram for the Evaluation phase.

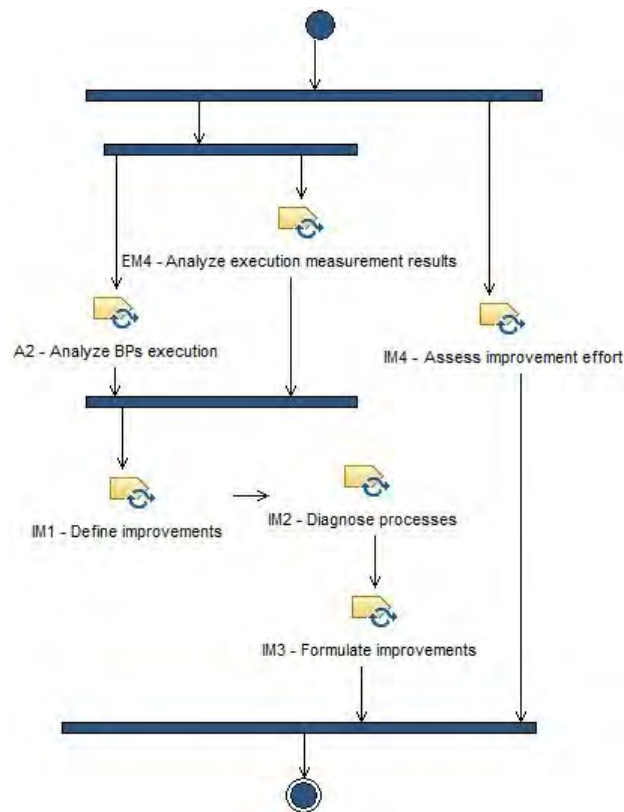


Figure 5.11.: BPCIP Evaluation phase activity diagram

## 5.4. EPF implementation

EPF Composer [Eclipse, 2004-2011] provides a way of defining, managing and reusing development processes specified using the OMG standard Software & Systems Process Engineering Metamodel (SPEM2) [OMG, 2008e]. SPEM2 allows software engineering processes to be represented, by defining the elements needed to specify a software process, such as disciplines, activities, artifacts, roles and lifecycle. EPF Composer is a graphical editor to define software processes using SPEM2, allowing the integration and interoperability of all processes specified in this way.

BPCIP, although is not a software engineering process is a continuous improvement process which can also be represented in this way, as presented below. It is also published as a web site by

means of the EPF Composer facilities, thereby giving easy access to it; it may also be published in the organization from the method plug-in provided, so it can be easily used. In Appendix B the complete BPCIP Web site is presented.

### 5.4.1. BPCIP method plug-in

The BPCIP method plug-in defines the BPCIP elements in the EPF Composer presented previously. The method content defines the content packages for the disciplines and their component elements defined for the disciplines, roles, tasks and guidance. The standard and custom categories allow the categorization of the defined method content by associating the elements with existing and new defined categories such as disciplines, role sets or work products. The processes enable the desired grouping of activities and tasks to be defined into process templates that can be reused to define the specific delivery process. In Figure 5.12a screenshot of the definition of BPCIP in the EPF Composer is shown.

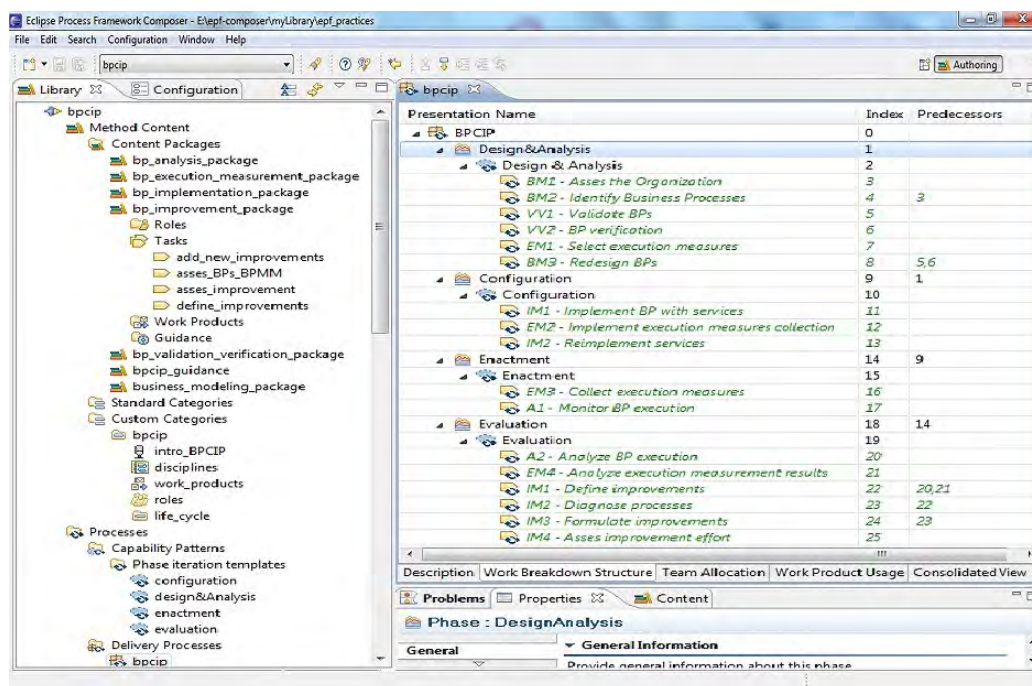


Figure 5.12.: Example of BPCIP method plug-in definition in EPF Composer

On the left side the tree of elements defined can be seen: method content with categories and elements, and processes with templates and delivery process. On the right side, among other information, there is the definition of the delivery process, with the phases defined, the activities included in each phase, and the defined sequence of activities. When all the elements are defined the export option allows BPCIP to be exported as a method plug-in, and the publish option makes it possible to publish BPCIP as a web site.

### 5.4.2. BPCIP web site

The BPCIP improvement process is easily accessed through the MINERVA Web site<sup>2</sup> where it is published, providing easy access and use of the elements defined. In Figure 5.13 a screenshot of the BPCIP web publication is shown, on the left side the defined categories can be seen: Disciplines, Work products, Roles and Lifecycle, along with some of the component elements such as activities, tasks, deliverables, roles. On the right side an example of task definition is presented for EM4

<sup>2</sup><http://alarcos.esi.uclm.es/MINERVA/BPCIP/Published/>

- Analyze execution measurement results, showing participating roles, work products defined as inputs and outputs, purpose, description and the Discipline to which it belongs.

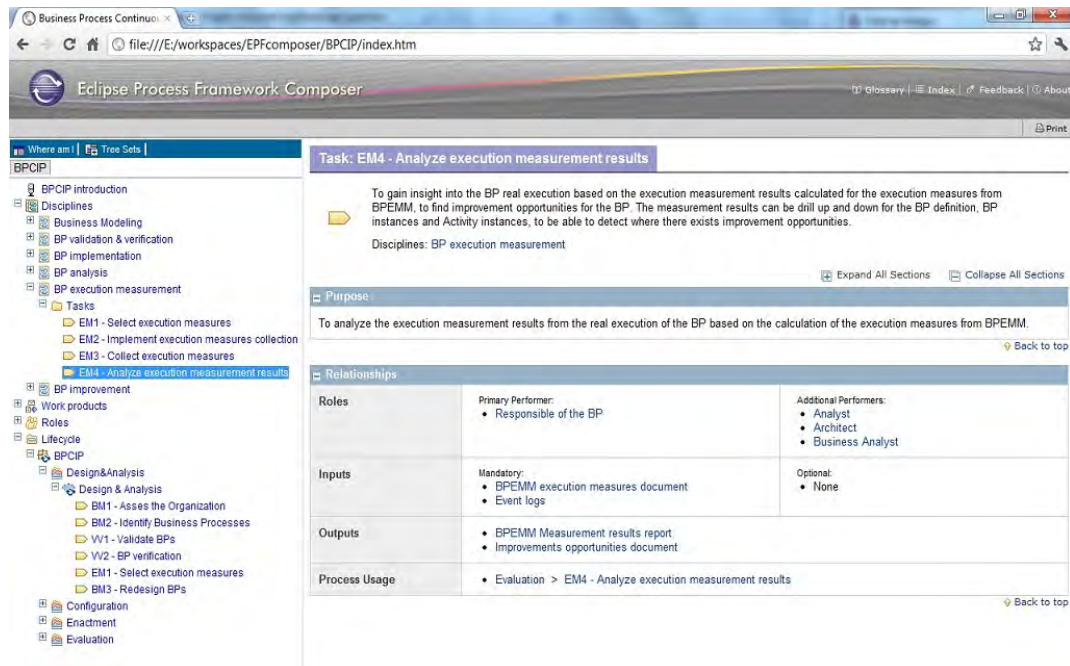


Figure 5.13.: Global view of BPSOM web site created using EPF composer

## 5.5. Conclusions

In this chapter the BPCIP continuous improvement process of MINERVA has been presented, which provides a systematic way for integrating improvements into the BPs of the organization. It defines a lifecycle to guide improvement efforts in organizations by extending the BP lifecycle [Weske, 2007] with explicit execution measurement and improvement activities. This provides the basis for measuring BPs execution and analyzing that, to find improvement opportunities that are then integrated into BPs using the improvement guide. The measurement activities include the use of the BPEMM execution measurement model described in chapter 6.

The Disciplines defined by BPCIP have been presented along with the defined activities, input and output artifacts and participating roles. For each Discipline a detail activity diagram is shown summarizing this information; a detailed description of each activity and its associated elements has also been provided. In addition, BPCIP phases making up the defined extended BP lifecycle of MINERVA have been presented, providing an activity diagram for each one that gives a general view of the flow between activities in each phase.

Finally, the implementation of the BPCIP as an EPF Composer method plug-in, which allows the integration and use of BPCIP in organizations in an easy and effortless way, has been explained. The corresponding Web Site generated from it, which provides easy access for its use throughout the organization has also been described.



## Chapter 6.

# Business Process Execution Measurement Model (BPEMM)

This Chapter describes the Business Process Execution Measurement Model (BPEMM) that has been defined in MINERVA framework to provide a set of execution measures that constitutes the basis for the execution measurement of BPs, in the context of the continuous improvement of BPs by means of BPCIP.

The Chapter is organized as follows: in section 6.1 a description of the BPEMM is presented, in section 6.2 the definition of BPEMM is presented including the different elements it is comprised of. In section 6.3 the execution measures that are integrated in the BPEMM are presented and in section 6.4 an example of the use of some execution measures of BPEMM is provided, finally in section 6.5 conclusions for the chapter are discussed.

The contents of this chapter complement the contents in chapter 5 which presents the Business Process Continuous Improvement Process (BPCIP) defining the extended BP lifecycle, including the execution measurement activities added in which BPEMM is used.

### 6.1. Introduction

The Business Process Execution Measurement Model (BPEMM) is defined in the methodological dimension of MINERVA, specifically it is integrated into BPCIP to support the improvement effort in the organization, providing a set of execution measures for measuring BPs execution that aim to relate it with the business and BP goals defined by the organization. Several execution measures for BPs, which are used to provide information about different aspects of BPs execution, are already defined in the literature as presented in chapter 3.

The main use of BPEMM in the context of BPCIP is in the execution measurement activities added to the BP lifecycle, to support the focus on BPs execution measurement. BPEMM is defined then to be used: by business people (1) to select the execution measures they want to obtain from each BP execution based on the predefined set of Goals it provides, by developers (2) to implement the selected measures into the BPs and services implementation, and (3) to collect the needed data from their execution, and by business people (4) to analyze the measurement results that are provided based on the data collected, as shown in Figure 6.1.

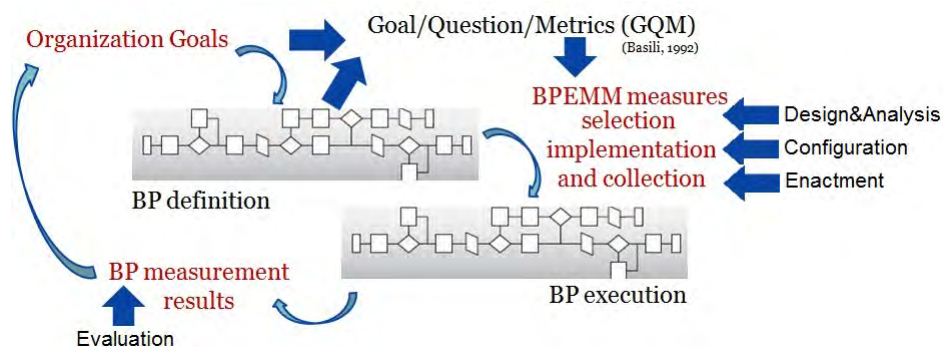


Figure 6.1.: BPEMM use in BPCIP

To provide traceability from the execution measurement results to the defined goals in the organization BPEMM has been defined based on the Goal, Question, Metrics (GQM) [Basili, 1992] paradigm, which allows business Goals for the organization to be defined, Questions to define how each goal will be evaluated to be formulated, and Metrics to answer the questions to be established. Execution measures in BPEMM are specified using the Software Measurement Ontology (SMO) [García et al., 2005] specifically differentiating into base measures, derived measures and indicators, which provides the basis for the implementation of the defined base measures into the execution of the BPs and services, and the calculation of the derived measures and indicators from the data registered.

BPEMM is defined specifically to measure the execution of BPs implemented by services, so execution measures in BPEMM are grouped into three views: Generic BP execution, Lean BP execution and Services execution, where the first groups together measures that can be applied to any kind of BP, the second includes measures to detect elements such as rework loops, and the third includes specific measures for services execution. Several existing execution measures have been evaluated and integrated into BPEMM based on the review of bibliography we have carried out using the advise of experts from the Business Intelligence (BI) area.

Other elements have been integrated in the definition of BPEMM such as the “devil’s quadrant” dimensions [Brand and van der Kolk, 1995, Reijers, 2003] of time, cost, flexibility and quality, to organize the measures inside each view according to these dimensions. A three level hierarchy for measures has been defined regarding BPs execution: Activity instances, each BP case and all BP cases for the BP model.

BPEMM is integrated in the method of work defined for the MINERVA framework for the BP-CIP lifecycle, as presented in chapter 4, to guide the continuous improvement effort based on BPs implemented by services driven by models. In Figure 6.2 the method of work of MINERVA framework as presented in chapter 4 is shown, to give the reader the context of the definitions that are described below. The definition of BPEMM is highlighted to show the focus of this chapter.

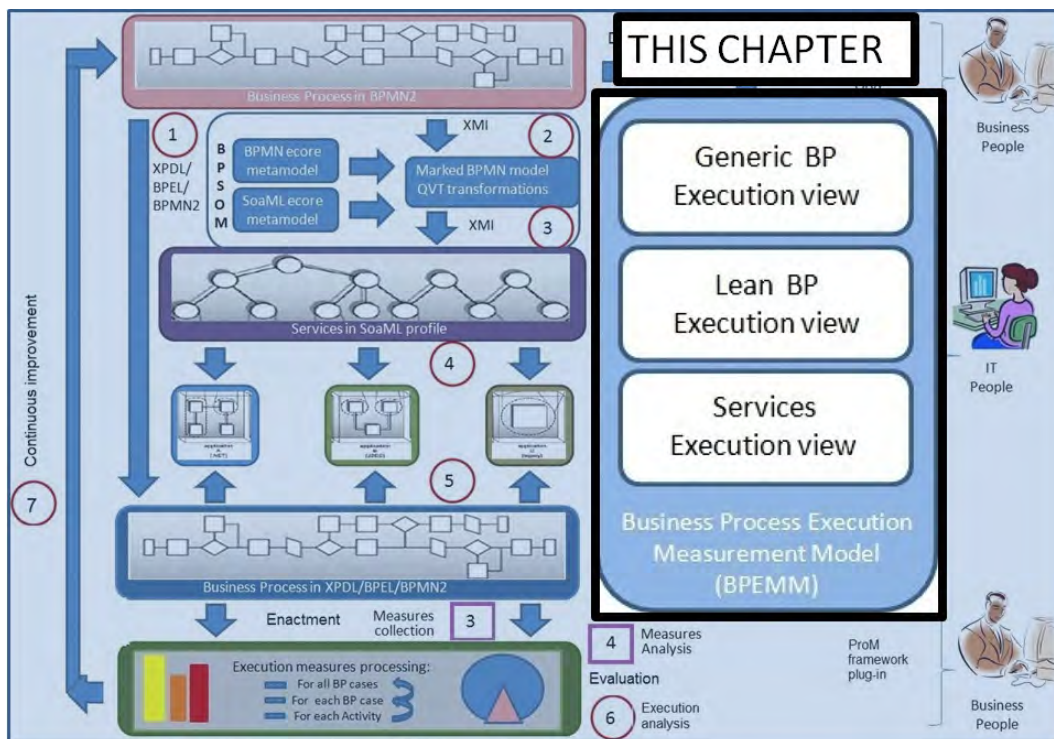


Figure 6.2.: MINERVA framework method of work



## 6.2. BPEMM definition

In this section the definition of BPEMM is shown presenting in the first place how the execution measures are defined and specified in BPEMM, secondly presenting the grouping and organization of the execution measures by means of views, dimensions and levels, and finally discussing some assumptions we defined for the calculation of the execution measures.

### 6.2.1. Execution measures specification

BPEMM has been defined based on the Goal, Question, Metrics (GQM) [Basili, 1992] paradigm, which is based on the idea that an organization must first specify its goals if it is to measure what the organization does in a meaningful way. It helps establish the necessary links between the business area that defines the business goals of the organization and BPs, and the measures to provide quantitative information on the compliance of BPs execution to the defined goals.

GQM integrates goals with process models, products, resources and different perspectives, depending on the needs of the organization and project. Initially defined to evaluate defects in software projects, its use has been expanded to several other domains such as improvement efforts in software organizations, and design of Software Engineering experiments. As our proposal includes several elements that also comes from the software area such as the improvement activities added to the BP lifecycle and BPMM which has its origins in CMMI and CMM as presented in chapter 3, the use of GQM to define BPEMM is set in the same direction.

The aim of BPEMM is to facilitate the selection of predefined execution measures by business people, and was designed specifically for BPs implemented by services. BPEMM measures are then specified applying the following concepts:

- **Goal:** defined for the organization, section, project or process, from various points of view and models.
- **Question:** describe how each goal will be evaluated from the point of view of a quality characteristic.
- **Metric:** a set of data, which can be objective or subjective, defined to answer each question quantitatively.

and also by means of the following elements defined in the Software Measurement Ontology (SMO) [García et al., 2005]:

- **Base measure:** a measure of an attribute with no dependence upon any other measure, and whose measurement approach is a measurement method.
- **Derived measure:** a measure derived from other base and derived measures using a measurement function as measurement approach.
- **Indicator:** a measure derived from other measures whose measurement approach is an analysis model which has an associated decision criteria (defining ranks to which the measurement results can belong).

In order to make clear the structure defined for the specification of execution measures in BPEMM in Table 6.1 an example is shown for the Throughput Time (TT) of the BP, but it should be borne in mind that this is not the actual and complete definition of measures for TT which is presented in section 6.3, but is given only to show the way in which all measures are defined in BPEMM.

**Table 6.1.:** Example of execution measures specification in BPEMM

<b>Goal</b>	<b>G1</b>	<b>Minimize the Throughput Time (TT) of the BP</b>
<b>Question</b>	<b>Q1</b>	<b>which is the actual TT of the BP</b>
<b>Measures</b>	<b>M1 (base)</b>	Start time of an Activity (ST)
	<b>M2 (base)</b>	Completion time of an Activity (CT)
	<b>M3 (derived)</b>	Working time of an Activity (AWoT = CT - ST)
	<b>M4 (derived)</b>	Throughput Time of a BP case (BPTT = TWoT + TWaT)
	<b>M5 (indicator)</b>	Average BP Throughput Time for all BP cases (ABPTT = $\sum$ BPTT / Total BP cases) Decision criteria = Inverse Percentage DC
<b>Decision Criteria</b>	<b>Percentage DC:</b>	R1: $0 \leq TTI \leq L1$ ="LOW"=RED; R2: $L1 < TTI < L2$ ="MEDIUM"=YELLOW; R3: $L2 \leq TTI$ ="HIGH"=GREEN

As can be seen in Table 6.1 for each business Goal defined one or more Questions are asked to provide answers to the Goal, in the example there is only one but there can be as many as are necessary. For each Question several execution measures are specified by means of the SMO: the base measures corresponding to data that can be obtained directly from the execution event log, the derived measures which are defined using base and/or derived measures, and indicators which can be also defined based on the previous ones. Indicators also have ranks attached to their measurement results to which they can belong, providing information about the results obtained. We use the ProM metaphor of semaphores to assign colors to ranks with the following meaning: “Green” for OK, “Yellow” for Warning and “Red” for Stop (Problems).

Several execution measures for BPs and services realizing them have been specified in BPEMM in the context of this thesis work, for each measurable concept identified as of interest for the business area, which are shown in Table 6.2 for each of the defined Execution views. For each of these measurable concepts, several Goals are defined and several questions and execution measures are provided by BPEMM, by means of base, derived and indicator measures. The set of execution measures is based on existing definitions we have found in the relevant literature regarding BPs and service execution measurement, adding some new ones to extend the information to be provided by the measures.

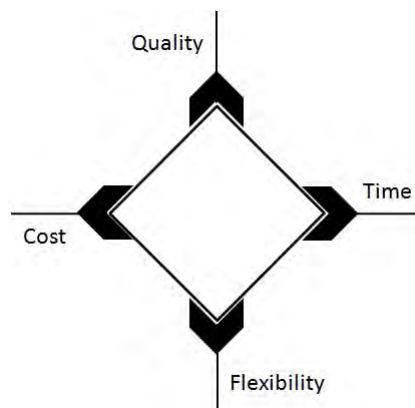
**Table 6.2.:** Measures and Goals defined by Execution View

<b>Execution view</b>	<b>measurable concept</b>	<b>Definition</b>	<b>Goals</b>
<b>Generic</b>	<b>Throughput Time (TT)</b>	Total time from the moment in which a BP case is initiated to its completion [Laguna and Marklund, 2005]	Min TT, Max efficiency
	<b>Capacity</b>	Number of BP cases per unit of time that the BP can handle, for resources (bottlenecks)[Laguna and Marklund, 2005] (adapted)	Max capacity, Min bottlenecks
	<b>Resources</b>	Tangible assets necessary to perform activities within a BP [Laguna and Marklund, 2005]	Min quantity, Max utilization
	<b>Cost</b>	Of human resources to produce a good or deliver a service (labor cost) [Reijers, 2003]	Min cost
	<b>Path execution</b>	Successful path execution (i.e. charging credit card) vs. unsuccessful path (i.e. charged rejected)	Max successful, Min unsuccessful
	<b>Final state</b>	State of BP ending, apart from successful or unsuccessful (i.e. normally, aborted, cancelled)	Max normal, Min abnormal
	<b>Quality</b>	External: user satisfaction with the product or process, internal: condition of working in BP [Reijers, 2003]	Max external and internal quality
	<b>Flexibility</b>	Ability to react to changes, i.e. resources executing tasks, process handling cases, change workloads, change structure [Reijers, 2003]	Max flexibility Min time for introducing changes

Execution view	measurable concept	Definition	Goals
Lean	<b>Rework</b>	Loop in the BP with control specifying criteria to allow a job to continue processing [Laguna and Marklund, 2005]	Min execution of rework loops
	<b>Value-adding activities</b>	Activities essential for the BP to meet customer's expectations [Laguna and Marklund, 2005]	Max value-adding activities
	<b>Non value-adding activities</b>	Activities that does not add value to the customer [Laguna and Marklund, 2005]	Min non value-adding activities
	<b>Defects / errors</b>	Defects/errors in process/products causes repair, rework and waste [Laguna and Marklund, 2005]	Min defects/ errors
Services	<b>Response Time</b>	Guaranteed time interval for the execution of the response of an event [Barbacci et al., 1995]	Guarantee a defined response time
	<b>Throughput</b>	Number of event responses completed over a given observation interval [Barbacci et al., 1995]	Guarantee a defined throughput
	<b>Capacity</b>	Maximum achievable throughput without violating specified responses time [Barbacci et al., 1995]	Guarantee a defined capacity
	<b>Availability</b>	Service readiness for usage [Barbacci et al., 1995]	Guarantee a defined availability
	<b>Reliability</b>	Service ability to keep operating over time [Barbacci et al., 1995]	Guarantee defined reliability
	<b>Confidentiality</b>	Property that data be inaccessible to unauthorized users [Barbacci et al., 1995]	Guarantee defined confidentiality
	<b>Integrity</b>	Property that the data be resistant to unauthorized modification [Barbacci et al., 1995]	Guarantee defined integrity

### 6.2.2. Views, dimensions and hierarchy

As mentioned above, BPEMM measures are grouped according to three defined views: Generic BP execution, Lean BP execution and Services execution, defining measures that can be applied to any kind of BP, to detect elements such as rework loops and for services execution, respectively, which are described in section 6.3. These views are organized taking into account the dimensions time, cost, flexibility and quality in the “Devil’s quadrant” [Brand and van der Kolk, 1995, Reijers, 2003] shown in Figure 6.3. The use of these dimensions help analyze the trade-offs that have to be taken into account when designing or redesigning a BP, since changes in one dimension can negatively impact another, i.e. removing an activity can improve the duration of the BP but can negatively affect its quality.



**Figure 6.3.:** Dimensions of the devil’s quadrangle [Brand and van der Kolk, 1995, Reijers, 2003]

Measures are also arranged in a three level hierarchy shown in Figure 6.4, which defines the Granularity level of the execution measures. At the third level measures for each activity instance are registered; in the second level these measures are combined to calculate measures for each corresponding BP case, and finally in the first level the BP case measures are combined to calculate the measures for the BP, such as averages or percentages, among others. Each level then, corresponds to the differentiated elements that are registered in BPs execution as presented in chapter 3: BP model, BP instances (cases) and activity model and its corresponding activity instances.

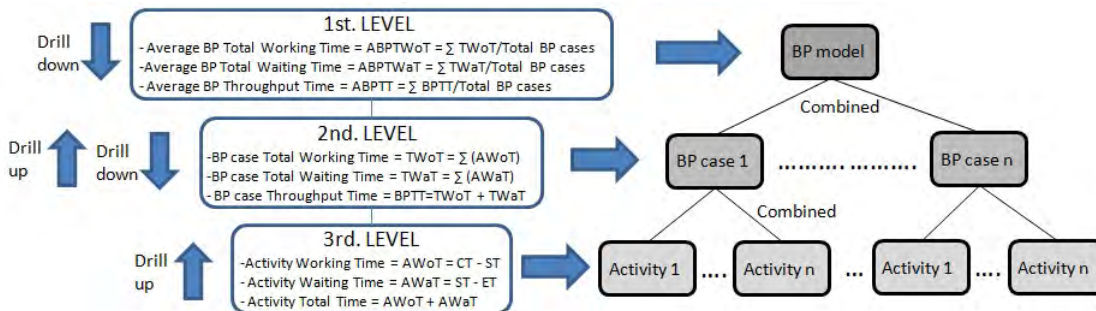


Figure 6.4.: Hierarchy of execution measures defined in BPEMM

As a graphical presentation of the tridimensional organization of BPEMM, we have defined a “cube” view in which the elements are shown: Execution views (Generic BP, Lean BP and Services), “Devil’s quadrant” dimensions (time, cost, flexibility and quality) and Granularity levels (BP, BP cases and Activity instances). These three dimensions are used to present the measurement results for several combinations of elements from the cube dimensions, by selecting a value in each dimension, so that the point  $m(x,y,z)$  defines the set of execution measures that corresponds to the intersection of selected values. In Figure 6.5 the cube is shown illustrating this vision.

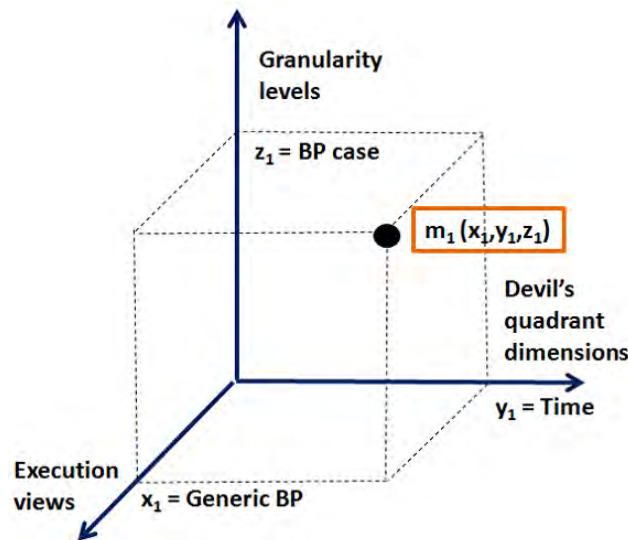


Figure 6.5.: Cube view of the execution measures defined in BPEMM

In the 3D-space presented in Figure 6.5 the example point  $m(x_1, y_1, z_1)$  represents the selection of: the Generic BP execution view from the Execution views dimension, the Time dimension from the dimensions of the Devil’s Quadrant dimension and the BP case from the Granularity levels dimension, which will result in the set of measures that can be calculated for the selection, for example, the Total working time of a BP case or the Throughput Time (TT) of a BP case.

### 6.2.3. Assumptions for calculations

In this Section we present some assumptions that are taken for the calculation of BPEMM execution measures. In the first place, regarding the activity lifecycle presented in chapter 3 and the several different states and transitions that can occur in a BP execution, we have simplified the approach defining a minimum core set of data that has to be collected in a mandatory way, to be able to calculate them. It corresponds to the three key times of: enabled, start and complete time, which we have decided to use as they are the ones most commonly provided by BP engines.

However, if the enabled time of the activities is not registered, we assume it to be the completion time of the previous activity, or in the case where the previous execution corresponds to a parallel branch, we assume it to be the completion time of the latest activity executed in the branches. Nevertheless the BPEMM execution measures can be easily extended to take into account other times that can be registered such as for the states suspended and resumed, which we will leave for future work. Data on the resource executing each activity is also required, so as to be able to calculate the cost of the BP associated with human resources.

Regarding states for the BP case, in general the start and completion times of BP cases are registered, but if they are not registered at all, they can be derived directly from the ones defined for the activities, since the execution state of an activity in a BP case determines the state in which the BP case is (i.e. if an activity in a BP case is in the state “In progress” then the corresponding BP case must be in the state “Started”), so we focus on activity times, states and transitions.

Regarding the type of loops that can be defined in the control flow of a BP model, we are interested only in the ones corresponding to a rework loop in the execution, which is defined when the controlling business rule refers to some quality aspects in the execution of the previous activities (for example examine a product to see if it has been repaired), which will determine if the flow can continue or it has to be re-executed in order to meet the levels defined for the execution. Another kind of loop can be defined to show that several activities can be executed more than once when, for example, buying goods on a web page, choosing an item to be added to the shopping cart each time, which does not represent rework but is a way of modeling the sequential selection of goods.

Regarding the BP model, we assume that each activity in the BP model will have a unique label, that is there are no repeated labels for the activities in a BP model. This allows us to define for example, which activities are involved in a rework loop by providing the label of the activity as identifier, or defining the “successful branch” of execution which will be the one presenting data for the execution of given activities. This occurs, for example, when paying by credit card for a successful or unsuccessful charge of the required amount. To be able to use this information for the calculation of the execution measures the BP model will be required. It will also be used to enable the times for the different constructions to be calculated, such as parallel branches and rework loops, as presented in chapter 3.

For the calculation of service execution times, several times were defined based on the enabled, start and completion times for activities as presented before, which will be explained in section 6.3 in the Services execution view, as there are several concepts and definitions needed to understand them and it is better to present the complete approach when defining the view and the corresponding execution measures.

All information that we need and can not be obtained from the collection of data from the BPs execution, will be provided in the form of a document or “configuration file”, and should be provided by the business area. This may include information such as the definition of rework loops or “successful branch”, or the salary of each participant in the BP to be able to calculate the cost associated with human resources participating in the BP, or the information on the number or resources assigned to each resource type in the BP model so as to be able to calculate the process capacity.

### 6.3. Execution measures

In this section we present the set of execution measures included in BPEMM classified in the three execution views we have defined: Generic BP, Lean focus and Services, and for each one the Devil's Quadrant dimensions of: time, cost, quality and flexibility are provided, when there are definitions for a particular one, and for the hierarchy defined.

#### 6.3.1. Generic BP execution view

In this view generic and domain specific execution measures for domains such as healthcare, software or production are integrated. These measures are related to generic BP characteristics that are not themselves related to the type of BP, such as duration of activities and of the complete BP, costs, roles involved, quality as perceived by the user. Domain specific measures refer to measures that have to be instantiated, in turn, for each domain, for example to measure "successful path" execution through the BP, specific activities included in the path have to be identified for each BP. This kind of measures is defined by the business area as Key Performance Indicators (KPI), taking into account specific expected results such as quantity of received or delivered orders, products, or successful payments. Execution measures for this view are presented below for each of the dimensions of time, cost, quality and flexibility.

##### 6.3.1.1. Time dimension

One key measure in this view refers to the **Throughput Time (TT)** [Reijers, 2003, Netjes, 2010, Laguna and Marklund, 2005, zur Muehlen, 2004], which as presented in chapter 3, is defined as the total time from the moment in which a BP case is initiated to its completion [Laguna and Marklund, 2005]. Recalling the definitions presented in chapter 3 the enabled time is that in which an activity becomes available for execution, the start time is the time in which the activity actually starts its execution and the completion time is when the activity completes its execution.

Based on these times we can calculate, among other things, the working and waiting time of an activity, the Throughput Time (TT) of a BP case and the average TT for all BP cases. In Figure 6.6 the relations between the times defined are shown, in a very simplified view as for the calculation of the TT for a BP case, it must be known which activities have to be added up from the BP model, in choosing, for example, which path to take into account in parallel branches, as explained in chapter 3.

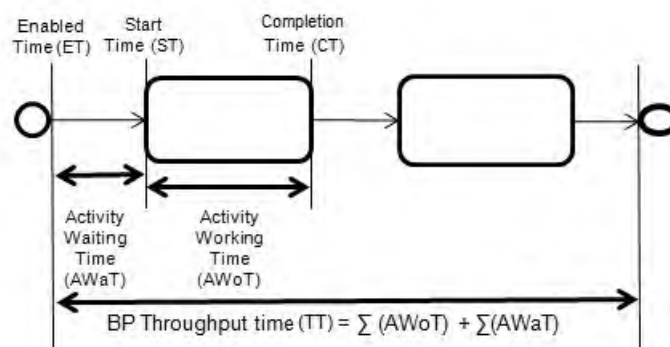


Figure 6.6.: Defined times for activities and BP instances execution

Several existing execution measures are integrated for the time dimension based on the ones defined in [Reijers, 2003, Laguna and Marklund, 2005, zur Muehlen, 2004], and some new ones are also defined such as the Index between Working and Waiting time for activities and BP cases (M10, M11), as shown in Table 6.3 where the measures defined for the time dimension - BP Throughput Time (TT) are shown.

**Table 6.3.:** Measures for Generic BP execution view & time dimension - Throughput Time (TT)

Goal	G1	Minimize the Throughput Time (TT) of the BP
Question	Q1	what is the actual TT of the BP
Measures	M1 (base)	Enabled time of an Activity (ET)
	M2 (base)	Start time of an Activity (ST)
	M3 (base)	Completion time of an Activity (CT)
	M4 (derived)	Working time of an Activity (AWoT = CT - ST)
	M5 (derived)	Waiting time of an Activity (AWaT = ST - ET)
	M6 (derived)	Total time of an Activity (ATT = AWoT + AWaT)
	M7 (derived)	Total Working time of a BP case (TWOt = $\sum$ (AWoT))
	M8 (derived)	Total Waiting time of a BP case (TwaT = $\sum$ (AWaT))
	M9 (derived)	Throughput Time of a BP case (BPTT = $\sum$ (ATT)) or (TWOt + TwaT) for the corresponding paths
	M10 (indicator)	Activity Working time vs. Activity Waiting time index (ATI = AWaT/AWoT) Decision criteria = Index DC.
	M11 (indicator)	Total BP Working time vs. Total BP Waiting time index (TTI =TwaT/TWOt) Decision criteria = Index DC.
	M12 (indicator)	Percentage of total BP Working time in total BP TT (PWOt = TWOt*100/BPTT) Decision criteria = Percentage DC.
	M13 (indicator)	Percentage of Total BP Waiting time in Total BP TT (PwaT = TwaT*100/BPTT) Decision criteria = Inverse Percentage DC
	M14 (indicator)	Average BP Throughput Time for all BP cases (ABPTT = $\sum$ BPTT / Total BP cases) Decision criteria = Inverse Percentage DC
	M15 (indicator)	Average BP total Working time for all BP cases (ABPTWOt = $\sum$ TWOt/Total BP cases) Decision criteria = Percentage DC
	M16 (indicator)	Average BP total Waiting time for all BP cases (ABPTwaT = $\sum$ TwaT /Total BP cases) Decision criteria=Inverse Percentage DC
	M17 (indicator)	Average Activity total Working time for all BP cases (AATWOt = $\sum$ AWoT / Number of BP cases in which the activity was executed) Decision criteria=Inverse Percentage DC
	M18 (indicator)	Average Activity total Waiting time for all BP cases (AATwaT = $\sum$ AWaT / Number of BP cases in which the activity was executed) Decision criteria=Inverse Percentage DC
	M19 (indicator)	Average Activity total time for all BP cases (AATT = $\sum$ ATT / Number of BP cases in which the activity was executed) Decision criteria=Inverse Percentage DC
Decision criteria	Index DC:	R1: $0 \leq TTI \leq L1$ ="LOW"=GREEN; R2: $L1 \leq TTI < L2$ ="MEDIUM"=YELLOW; R3: $L2 \leq TTI$ ="HIGH"=RED
	Percentage DC:	R1: $0 \leq TTI \leq L1$ ="LOW"=RED; R2: $L1 \leq TTI < L2$ ="MEDIUM"=YELLOW; R3: $L2 \leq TTI$ ="HIGH"=GREEN

The execution measures presented in Table 6.3 allow us to show information for each level of the hierarchy, as times are calculated for each activity, each BP case and averages for the BP model level. When analyzing the measurement results for this view and dimension a key element is to search for high waiting times and high percentages of waiting times in each of the levels, high average times for the TT of the BP, and high indexes between working and waiting times, which will indicate that for the level analyzed, for example an activity, the waiting time is much higher than the working time.

Another key measure for the time dimension is the **Capacity of the BP** which was also presented in chapter 3 and represents the number of BP cases that can be processed by unit of time, which is restricted by the resources associated to the roles executing the activities in the BP [Laguna and Marklund, 2005]. Recall that the capacity of the BP refers to the resource type and resources available for each one, so a key measure for capacity is to find the bottleneck of the BP i.e. the resource type with the smallest capacity.

The base measure defined for the BP capacity is the assignation of resources to roles executing each activity, which is not contained in the execution event log, but in the information we gather from the business area in the form of a document or context data to specify it; from the BPMN2 model the information about the role assigned to each activity is obtained. In Table 6.4 the BPEMM execution measures defined for the time dimension - BP Capacity, which we have adapted from [Laguna and Marklund, 2005], are presented. Note that we assume the equality of resource type = role (as defined by lanes in the BPMN2 model) in the definitions.

**Table 6.4.:** Measures for Generic BP execution view & time dimension - Capacity

Goal	G2	Maximize the capacity of the BP
Question	Q1	what is the actual capacity of the BP
Measures	M1 (base)	Number of resources per role defined in the BP = NRRBP (from context data)
	M2 (base)	Number of execution of each activity in all BP cases ( NEA = count the times the activity is executed in all BP cases execution)
	M3 (derived)	Number of jobs processed by each activity in the BP (NJA = NEA/Total BP cases)
	M4 (derived)	Unit load for a resource in the BP ( $ULR = \sum (AATW_oT*NJA)$ )
	M5 (derived)	Unit capacity for each resource (UCR = 1/ULR indicates the number of jobs each resource can complete per unit of time)
	M6 (derived)	Pool capacity for each role in the BP (PCR = UCR*NRRBP)
	M7 (derived)	Process capacity of the BP (PCBP = Bottleneck of the BP = smallest of measure 6)
	M8 (derived)	Throughput rate of the BP = arrival rate to the system corresponding to the average number of jobs eventually served per unit of time (TRBP = total of BP cases / number of time periods)
	M9 (derived)	Capacity utilization for a resource (CUR = TRBP / PCR)

Referring back to the definitions in chapter 3 these measures were adapted to be based on actual information on the BP execution, as opposed to probabilities or speculations on the number of resources for each resource type, which are used when simulating or analyzing the possible execution of BPs. Since from the registered data we know how many times each activity has been executed, and we also know the average working times for the activities of the BP, we can calculate the unit capacity of each resource type, which indicates the number of BP cases each one can complete per unit of time (with the unit of time of the average time of the activities: seconds, minutes, hours, etc.), and hence the resource type with the smallest capacity indicates the bottleneck of the process. Note that we also have information on the specific execution of activities for each resource assigned to each activity, but as the averages times are used here, it does not add any vital information, although it would be interesting for managers to see the averages times of execution for each activity assigned to each person, we leave this for future work.

### 6.3.1.2. Cost dimension

For the cost dimension we have integrated execution measures to calculate in the first place the operational cost, that is the cost associated with resources assigned to activities, which can be human resources or material resources such as an operation room. To be able to calculate the cost the information about the base cost of each resource must be provided in the desired unit of time (hour, week, month) so this can be used to calculate each participation in the BP. In Table 6.5 the execution measures defined for the cost dimension are shown.



**Table 6.5.:** Measures for Generic BP execution view & cost dimension

Goal	G3	Minimize the cost of the BP
Question	Q1	What is the actual cost of the BP
Measures	M1 (base)	Resource cost per unit of time = RCT (from context data)
	M2 (derived)	Cost per activity in a BP case ( $ACo = AWoT * RCT$ )
	M3 (derived)	Total cost per activity in all BP cases ( $TACo = \sum ACoBP_{(i)}$ )
	M4 (derived)	Total cost of BP case ( $TCo = \sum ACo_{(i)}$ )
	M5 (derived)	Total cost of BP for all BP cases ( $TBPCo = \sum TCo$ )
	M6 (indicator)	Percentage of activity cost in BP case ( $PACo = ACo * 100 / TCo$ ) Decision criteria = Cost DC
	M7 (indicator)	Percentage of activity cost in all BP cases ( $PTACo = TACo * 100 / TBPCo$ ) Decision criteria = Cost DC
	M8 (indicator)	Percentage of BP case cost in all BP cases ( $PTCo = TCo * 100 / TBPCo$ ) Decision criteria = Cost DC
	M9 (indicator)	Average cost of BP for all BP cases ( $ABPCo = TBPCo / \text{Total BP cases}$ ) Decision criteria = Cost DC
	M10 (indicator)	Average cost of activity for all BP cases ( $AACo = TACo / \text{Total BP cases}$ ) Decision criteria = Cost DC
Decision criteria	Cost DC:	R1: $0 \leq TTI \leq L1 = \text{"LOW"} = \text{GREEN}$ ; R2: $L1 < TTI < L2 = \text{"MEDIUM"} = \text{YELLOW}$ ; R3: $L2 \leq TTI = \text{"HIGH"} = \text{RED}$

As can be seen in Table 6.5 costs are calculated for each activity in the BP, for each BP case and for all BP cases, to provide a complete view covering the hierarchy defined. Although this is an initial set of measures they can provide valuable information on the costs associated with the BP, which we plan to extend as future work.

### 6.3.1.3. Quality dimension

For the quality dimension we provide information about the way in which the BP cases “end” their execution, with two different approaches: the first one corresponds to the type of ending as defined in the BP lifecycle as presented in chapter 3, such as completed, terminated and aborted. These measures provide information about the number of BP cases ending successfully or unsuccessfully, as well as the percentage they represent in the total BP cases execution. This will provide valuable information if for example it is found that less than half of the BP cases are ended in the complete state. These execution measures for the quality dimension by type of ending are shown in Table 6.6.

**Table 6.6.:** Measures for Generic BP execution view & quality dimension - Type of ending

Goal	G4	Maximize the number of BP cases ending normally
Question	Q1	What is the actual number of cases ending normally
Measures	M1(base)	Number of BP cases ending in the selected state = NBPE (count BP cases ending in states: COMPLETED, TERMINATED, ABORTED)
	M2 (indicator)	Percentage of BP ending in completed state in total BP cases ( $PBPCo = NBPE * 100 / \text{Total BP cases}$ ) for state = COMPLETED. Decision criteria = Percentage Completed DC
	M3 (indicator)	Percentage of BP ending in terminated state in total BP cases ( $PBPTe = NBPE * 100 / \text{Total BP cases}$ ) for state = TERMINATED. Decision criteria = Inverse Percentage Complete DC
	M4 (indicator)	Percentage of BP ending in aborted state in total BP cases ( $PBPCo = NBPE * 100 / \text{Total BP cases}$ ) for state = ABORTED. Decision criteria = Inverse Percentage Complete DC
Decision criteria	Percentage Comp DC:	R1: $0 \leq TTI \leq L1 = \text{"LOW"} = \text{RED}$ ; R2: $L1 < TTI < L2 = \text{"MEDIUM"} = \text{YELLOW}$ ; R3: $L2 \leq TTI = \text{"HIGH"} = \text{GREEN}$

The second one corresponds to the definition of the “successful path” that has to be instantiated for each BP and domain, by defining the activities included in the successful execution branches of the BP. This has to be done by the business people when selecting the execution measures from BPEMM, and all this is registered in a document or “configuration file” which will then be used for the calculation of the measures.

This approach is complementary to the first one presented, as it could be interesting to know for all the BP cases which ended in the complete state (previous measures) which of them were successful or unsuccessful, when for example charging a credit card for a sell. This information can be useful to detect for example a malfunction in the implementation of the BP, where many credit card charges are rejected due to poor interaction between services implementing the BP. In Table 6.7 the execution measures defined for the quality dimension regarding the successful branch execution are shown.

**Table 6.7.:** Measures for Generic BP execution view & quality dimension - Successful branch

<b>Goal</b>	<b>G5</b>	<b>Maximize the number of BP cases ending successfully (executes the successful branch of the BP)</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual number of BP cases ending successfully</b>
<b>Measures</b>	<b>M1 (base)</b>	Number of BP cases ending successfully or unsuccessfully = NBPBE (count BP cases with activities in the successful or unsuccessful branch as defined in the context data)
	<b>M2 (indicator)</b>	Percentage of BP ending successfully in total BP cases (PBPSB = $NBPBE * 100 / \text{Total BP cases}$ ) for successful branch. Decision criteria = Percentage Successful DC
	<b>M3 (indicator)</b>	Percentage of BP ending unsuccessfully in total BP cases (PBPUSEB = $NBPBE * 100 / \text{Total BP cases}$ ) for unsuccessful branch. Decision criteria = Inverse Percentage Successful DC
<b>Decision criteria</b>	<b>Percentage Successful DC:</b>	R1: $0 \leq TTI \leq L1 = \text{"LOW"} = \text{RED}$ ; R2: $L1 < TTI < L2 = \text{"MEDIUM"} = \text{YELLOW}$ ; R3: $L2 \leq TTI = \text{"HIGH"} = \text{GREEN}$

Other measures for quality that are not presented here can include measuring Client satisfaction and Internal people satisfaction with questionnaires about their perception on the BP execution, which can also lead to an improvement in the way things are done. The definition of execution measures for this dimension will be extended in future work.

#### 6.3.1.4. Flexibility dimension

We have no measures defined yet as part of BPEMM for the flexibility dimension, as we leave this for future work. These are related to, for example, the times regarding the “change process” for the BP, which may originate from different sources: a request to change something in the BP by any people involved in its execution, an unforeseen event that prevents them from following the BP model as defined, or an improvement opportunity detected by business people in the evaluation phase by analyzing the BPEMM execution measure values.

In this dimension what we want to measure is for example, how long it will take to integrate the needed change into the BP, for which we will define a change process which will be modeled, implemented and executed as with any other BP in the organization. This process will be used by business people to track the changes made to the BP and the duration of the change process. So the flexibility execution measures defined are built upon the execution measures of the time and quality dimension, but with a focus on the particular BP of introducing changes into any other BP in the organization.

### 6.3.2. Lean execution view

This view defines execution measures to collect information for detecting waste in the BP execution. It aims to find activities, paths or parts in the BP that if optimized can lead to an optimization and improvement of the complete BP [Laguna and Marklund, 2005]. Lean thinking was first introduced in the Toyota Production System (TPS) and is based on the identification and elimination of waste, which is categorized in seven types: overproduction, waiting, transport, extra processing, inventory, motion and defects.

Lean principles and waste types have been adapted to areas other than the manufacturing one, such as software development [Poppendieck, 2002], information management [Hicks, 2007] and healthcare [Jimmerson et al., 2005]. Non value-adding activities are usually related to handoffs, delays, rework, and control activities in loops. Measures in this view are closely related to each other, because when a loop is defined in the flow of the BP, a control activity is usually added to determine if execution can continue or must go back, with respect to the control definitions.

For this view the most important dimensions are the time (focused on rework loops) and the quality dimension; the cost and flexibility dimension are the same as the ones defined for the Generic BP view.

#### 6.3.2.1. Time dimension

The rework loop allows for the identification of non-value adding activities that generate delays, so we focus the definition of lean execution measures in the time dimension on the discovering of the level of rework in a BP by means of the times a rework loop represents in the BP. From the time dimension of the Generic BP view base and derived measures defined are used here in the calculation of some of the rework measures. In Table 6.8 the execution measures defined for the rework in the Lean execution view are shown.

**Table 6.8.:** Measures for Lean execution view & quality dimension

Goal	G1	Minimize the rework in loops of the BP
Question	Q1	What is the actual quantity of rework due to BP loops
Measures	M1 (base)	Number of executions of an activity in a rework loop = NARL (counts the times each activity is executed in a rework loop as defined in the context data)
	M2 (derived)	Activity Working time for the rework in a loop ( $AWoTRL = \sum AWoT(e_i)$ being $e_i$ each execution of the activity in the loop)
	M3 (derived)	Total Working time for the rework in a loop of the BP ( $TWoTRL = \sum AWoTRL(a_i)$ where $a_i$ represents an activity in the loop)
	M4 (derived)	Total Working time for rework in all loops of BP case ( $BPTWoTRL = \sum TWoTRL(l_i)$ where $l_i$ represents a loop in the BP)
	M5 (derived)	Total Working time for rework of an activity in all BP cases ( $TAWoTRL = \sum AWoTRL$ )
	M6 (derived)	Total of BP cases with execution of rework loops ( $TBPERL =$ counts the BP cases with execution of rework loops)
	M7 (indicator)	Percentage of rework time in BP case due to loops in the total BP TT ( $PBPTWoTRL = BPTWoTRL*100/BPTT$ ) Decision criteria = Percentage DC
	M8 (indicator)	Percentage of BP cases with execution of rework loops ( $PTBPERL = TBPERL*100/Total\ BP\ cases$ )
	M9 (indicator)	Percentage of rework time for an activity due to execution of rework loops in all BP cases ( $AAWoTRL = TAWoTRL*100/$ Number of BP cases in which the activity was executed)
Decision criteria	Percentage DC:	R1: $0 \leq TTI \leq L1 = "LOW" = GREEN$ ; R2: $L1 < TTI < L2 = "MEDIUM" = YELLOW$ ; R3: $L2 \leq 100 = "HIGH" = RED$

Complementary to the measures presented in Table 6.8, a key aspect for the lean vision of BP execution is to measure the progress of the BP against the target duration defined for it. This involves using the TT execution measures defined in the Generic BP execution view for the time dimension, to position each BP case execution in the time line that shows at which stage of the BP completion it is at a given moment. In Lean it is also important to measure the relationship with the providers of the organization, to be able to detect the impact of the providers' delays in the execution times of the BPs in the organization. The times for the providers' execution can be measured as activities or sub-process times in the BP, when they relate to providers.

### **6.3.2.2. Quality dimension**

Other measures for Lean involve the discovering of defects during the BP execution, which can be done by any worker at any time during the execution of the BP, informing and even cancelling the BP case if the error is not recoverable for that unit or service being delivered. This information is registered as part of the quality dimension execution measures regarding the type of ending of the BP, for example when a BP is cancelled by the user a brief description of the problem detected can be required by the system. We will add specific measures to complete this dimension as future work.

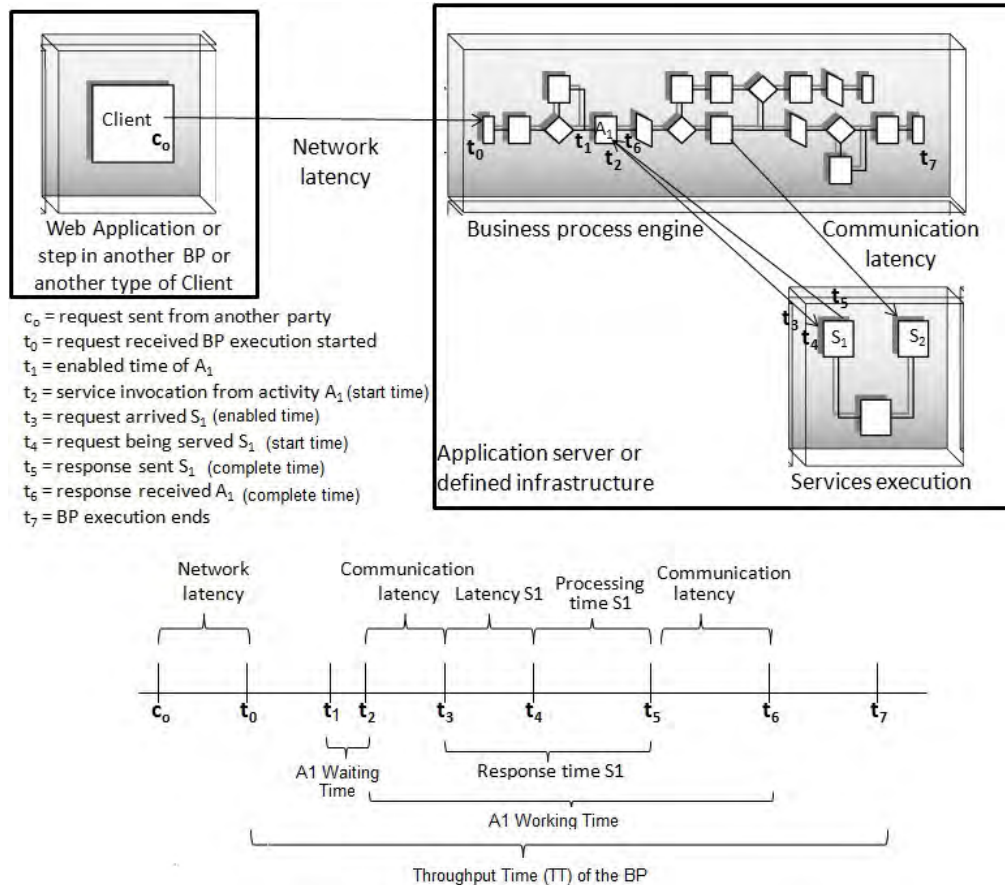
### **6.3.3. Services execution view**

This view contains measures regarding the execution of services realizing the BPs, taking into account the Quality of Services (QoS) requirements for this type of software. Services measures are based on quality attributes for services such as: performance (i.e. response time: processing time and latency, throughput, capacity), security (i.e. confidentiality, integrity), dependability (i.e. availability, reliability), as defined in [Barbacci et al., 1995, O'Brien et al., 2005, Clements et al., 2001, Bass et al., 2003, Sahai et al., 2001, Cardoso et al., 2002] and organized by means of the "devil's quadrangle" dimensions.

#### **6.3.3.1. Time dimension**

To calculate the measures corresponding to the services execution view in the time dimension we have defined six times of interest for the activities and services execution to be registered in the BP execution. In the first place, in the BP activity we log the defined enabled time, the start time corresponding to when a service is invoked, and the complete time corresponding to when the service returns an answer after processing the request. In the second place, in the service itself we log the times in which the service received the invocation (enabled), starts its execution (start), completes its execution (complete) and sends the result to the BP engine, which are shown in Figure 6.7.

To calculate the measures for each service being invoked, the data about the invocation has to be logged in the service, which can be included for example in the application server running the service, such as log the time of the invocation, origin and credentials in every invocation to the service. When the service is invoked outside the organization and we do not have control on the environment in which it is executing, we can use information from the communication, but the internal times for the service will be difficult to obtain.



**Figure 6.7.:** Defined times for services execution and BP activities

As shown in Figure 6.7 the time  $t_1$  is used to log the enabled time of the activity invoking the service,  $t_2$  and  $t_6$  to log in the activity the invocation and answer received from the service respectively. This is done in the same way as if they were start and completion times as defined before, the only difference being that the resource executing it is the system; and  $t_3$ ,  $t_4$  and  $t_5$  to register times in the service: the invocation received (enabled =  $t_3$ ), the start =  $t_4$  and the complete =  $t_5$  times of the services processing. When we do not have the complete data on services execution, we can nevertheless use the times registered from the point of view of the BP, to calculate the response time of the service which will include communication latency, as well as latency and processing times for the service.

In this way, we are able to log in the BP engine the start and complete times for the activity in the same way we do with the rest of the activities in the BP, calculating the waiting and working times of the activity, which include the times for the service execution. With this approach, services execution times are also included in the calculation of the Throughput Time (TT) of the BP, and can also be shown apart only for those activities that involve the invocation of a service, providing the services measures for the BP implementation inside the organization, or evaluating the interactions with partners.

As discussed before, several base measures have to be included in the implementation of services or in the infrastructure executing them, so what we provide in this view is the data that has to be logged, so that when the implementation is done, it can be added to the services. In Table 6.9 the services execution measures defined for the time dimension - Service Response Time are shown.

**Table 6.9.:** Measures for Service execution view & time dimension - Service Response Time

<b>Goal</b>	<b>G1</b>	<b>Guarantee (average) service response time to (L1) seconds (L1 label to be changed)</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual (average) response time of the service</b>
<b>Measures</b>	<b>M1 (base)</b>	Invoke time of a service from the activity in the BP (IT = timestamp)
	<b>M2 (base)</b>	Enabled time of a service (ET = timestamp)
	<b>M3 (base)</b>	Start time of a service (ST = timestamp)
	<b>M4 (base)</b>	Completion time of a service (CT = timestamp)
	<b>M5 (base)</b>	Failed time of a service (FT = timestamp)
	<b>M6 (base)</b>	Answer time from the service to the activity in the BP (AT = timestamp)
	<b>M7 (derived)</b>	Service processing time (SPoT = CT - ST)
	<b>M8 (derived)</b>	Service latency time (SLaT = ST - ET)
	<b>M9 (derived)</b>	Service response time (SRpT = SPoT + SLaT)
	<b>M10 (derived)</b>	Service answer time from the BP (SAnT = AT - IT)
	<b>M11 (indicator)</b>	Service Processing time vs. Service Latency time index (STI = SLaT/SPoT) Decision criteria = Index DC
	<b>M12 (indicator)</b>	Average service response time in all BP cases (ASRpT = $\sum$ (SRpT / Total service executions in all BP cases) Decision criteria = Index DC
	<b>M13 (indicator)</b>	Average service answer time in all BP cases (ASAnT = (SAnT / Total service executions in all BP cases) Decision criteria=Index DC
<b>Decision criteria</b>	<b>Index DC:</b>	R1: $0 \leq TTI \leq L1 = "LOW" = GREEN$ ; R2: $L1 < TTI < L2 = "MEDIUM" = YELLOW$ ; R3: $L2 \leq TTI = "HIGH" = RED$

Other measures for services execution times correspond to the service throughput, which refers to the number of event responses completed over a given observation interval, and can only be calculated from the implementation of the service or the service infrastructure, so they only make sense for internal services in the organization. In Table 6.10 the services execution measures defined for the time dimension - Service Throughput are shown.

**Table 6.10.:** Measures for Service execution view & time dimension - Service Throughput

<b>Goal</b>	<b>G2</b>	<b>Guarantee service throughput to (S) service execution completed per period (P1) (S and P1 labels to be changed)</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual service throughput S1 service execution completed over the period P1</b>
<b>Measures</b>	<b>M1 (base)</b>	Number of S1 services execution over the period P1 = NSEOP (count services execution COMPLETED, FAILED or IN PROGRESS in the period P1)
	<b>M2 (indicator)</b>	Percentage of S1 service execution completed over the period P1 (PSECP = $NSEOP * 100 / \text{Total services execution including in progress}$ ) Decision criteria = SE completed DC
	<b>M3 (indicator)</b>	Percentage of S1 service execution failed over the period P1 (PSEFP = $NSEOP * 100 / \text{Total services execution including in progress}$ ) Decision criteria = Inverse SE completed DC
<b>Decision criteria</b>	<b>SE completed DC:</b>	R1: $0 \leq TTI \leq L1 = "LOW" = RED$ ; R2: $L1 < TTI < L2 = "MEDIUM" = YELLOW$ ; R3: $L2 \leq TTI = "HIGH" = GREEN$

There are other measures for services execution times which are related to the previous ones, throughput and response times, and correspond to the service capacity, which refers to the maximum achievable throughput without violating specified response times. This too can only be calculated from the implementation of the service or the service infrastructure also, so they only make sense for internal services in the organization. In Table 6.11 the services execution measures defined for the time dimension - Service Capacity are shown.

**Table 6.11.:** Measures for Service execution view & time dimension - Service Capacity

<b>Goal</b>	<b>G3</b>	<b>Guarantee service capacity to (S) service execution maintaining the (L1) seconds defined for service response time (S and L1 labels to be changed)</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual service capacity</b>
<b>Measures</b>	<b>M1 (base)</b>	Number of S1 service execution completed in $\leq L1$ seconds over the period P1 ( $NSECLP = \sum SECLP$ in the period P1)
	<b>M2 (base)</b>	Number of S2 service execution completed in $L2 > L1$ seconds violating agreements over the period P1 ( $NSECVLP = \sum SECVLP$ in the period P1)
	<b>M3 (base)</b>	Number of S3 service execution in progress in $L2 > L1$ seconds violating agreements over the period P1 ( $NSEIPVLP = \sum SEIPVLP$ in the period P1)
	<b>M4 (indicator)</b>	Service capacity ( $SCA = NSECLP*100 / NSECLP + NSEIPVLP+NSECVLP+NSEFP$ ) Decision criteria = Percentage SCA DC
	<b>M5 (indicator)</b>	Service capacity violation rate ( $SCVR = NSECVLP+NSEIPVLP*100 / NSECLP+NSEIPVLP+NSECVLP+NSEFP$ ) Decision criteria = Inverse Percentage SCA DC
<b>Decision criteria</b>	<b>Percentage SCA DC:</b>	R1: $0 \leq TTI \leq L1 = "LOW" = RED$ ; R2: $L1 < TTI < L2 = "MEDIUM" = YELLOW$ ; R3: $L2 \leq TTI = "HIGH" = GREEN$

### 6.3.3.2. Quality dimension

Most of the services execution measures data can only be gathered by implementing the log of the base measures in the services and/or the infrastructure. For example, to know how many unauthorized attempts to invoke a service have taken place over a certain period of time, we need to log all the invocations that were rejected due to their presenting invalid credentials to the service. For this and related goals, the service execution measures defined for the quality dimension are shown in Table 6.12, regarding availability, reliability and confidentiality.

**Table 6.12.:** Measures for Service execution view & quality dimension

<b>Goal</b>	<b>G1</b>	<b>Guarantee (A1) availability for the service (A1 label to be changed) - Dependability</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual availability of the service</b>
<b>Measures</b>	<b>M1 (derived)</b>	Service down time ( $SDT = ET - FT$ being ET the time when the service is back up)
	<b>M2 (derived)</b>	Total service down time over the period P1 ( $TSDT = \sum SDT$ in the period P1)
	<b>M3 (indicator)</b>	Service Availability over the period P1 ( $SA = P1 - TSDT / P1 * 100$ ) Decision Criteria = Percentage SR DC
<b>Goal</b>	<b>G2</b>	<b>Guarantee (R1) reliability for the service (R1 label to be changed) - Dependability</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual reliability of the service</b>
<b>Measures</b>	<b>M1 (base)</b>	Number of service execution initiated over the period P1 = NSEIP (counts the services ST initiated in the period P1)
	<b>M2 (indicator)</b>	Service Reliability ( $SR = NSRECP / NSEIP * 100$ ) Decision criteria = Percentage SR DC
<b>Goal</b>	<b>G3</b>	<b>Guarantee (C1) confidentiality level for the service (C1 label to be changed) - Security</b>
<b>Question</b>	<b>Q1</b>	<b>What is the actual confidentiality level of the service</b>
<b>Measures</b>	<b>M1 (base)</b>	Number of service invocations rejected due to invalid credentials over the period P1 = NSIR (counts the service invocations rejected in the period P1)
	<b>M2 (indicator)</b>	Percentage of service invocations rejected in all services invocations over the period P1 ( $PSIRSI = NSIR * 100 / NSIR + NSEIP$ ) Decision criteria = Inverse Percentage SR
<b>Decision criteria</b>	<b>Percentage SR DC:</b>	R1: $0 \leq TTI \leq L1 = "LOW" = RED$ ; R2: $L1 < TTI < L2 = "MEDIUM" = YELLOW$ ; R3: $L2 \leq TTI = "HIGH" = GREEN$

## 6.4. Example of application

To illustrate the use of some of the BPEMM execution measures an example of its application is presented in this section. Although the measures will be calculated automatically based on the data collected from BPs execution in the execution event logs, we present here how the data is used for the calculation of a BP case Throughput Time (TT), Capacity and Cost corresponding to the Generic BP view, time and cost dimensions.

As the calculations presented are mainly the base and derived ones for the activity and BP case levels, they are very simple, but the objective of the example is only to illustrate their use. The calculation of the rest of the measures is not presented as several pieces of data for execution are needed to be able to show them properly.

The BP chosen for the example is the “Patient Admission and Registration for Major Ambulatory Surgery (MAS)” from the Hospital General de Ciudad Real (HGCR) project on which we are working, simplified and adapted to be used as example, which is shown in Figure 6.8.

The MAS Patient requests an appointment for MAS from the Local Public Hospital; the Hospital’s Secretary then assigns the date and time, and sends the information back to the patient, simultaneously requesting the Patient’s medical record from the Central Health Register. On the assigned date, the patient arrives at the Hospital and presents the doctor’s surgery order, preconditions for the surgery are checked (for example, blood tests), and if any problem is found the Patient is informed and the surgery is cancelled, otherwise the established MAS preparation actions take place.

### 6.4.1. Calculations for the BP generic view

In this section the calculation of some of the BPEMM execution measures defined in the BP generic view are presented, for the dimensions: time (showing Throughput Time (TT) and capacity measures), and cost.

#### Time dimension, Throughput Time (TT)

The calculation of some execution measures as presented in section 6.3 is based on execution event logs in which data about BPs execution was collected. As these files can be very long and to show meaningful data for the example for various BP cases we will need a considerable part of the file, so we only present an example of an execution event log for arbitrary BP cases 1 and 2 in Table 6.13 with columns BP case, activity, timestamps and corresponding event (start, complete) and originator. Then we present data for another execution of a BP case in Figure 6.9, for the activities in the HGCR pool, as this is the organization in which the BP executes.

**Table 6.13.:** Example event log for the Patient MAS BP

BP case	Activity	TimeStamp	Event	Originator
1	Receive request for appointment	10-01-2010: 09:30	Start	Juan
1	Receive request for appointment	10-01-2010: 09:50	Complete	Juan
2	Receive request for appointment	10-01-2010: 09:35	Start	María
2	Receive request for appointment	10-01-2010: 09:45	Complete	María
2	Assign date for surgery	10-01-2010: 09:55	Start	Eva
1	Assign date for surgery	10-01-2010: 10:00	Start	María
1	Assign date for surgery	10-01-2010: 10:10	Complete	María
1	Send assigned date for surgery	10-01-2010: 10:15	Start	Juan
2	Assign date for surgery	10-01-2010: 10:15	Complete	Eva
1	Send assigned date for surgery	10-01-2010: 10:20	Complete	Juan



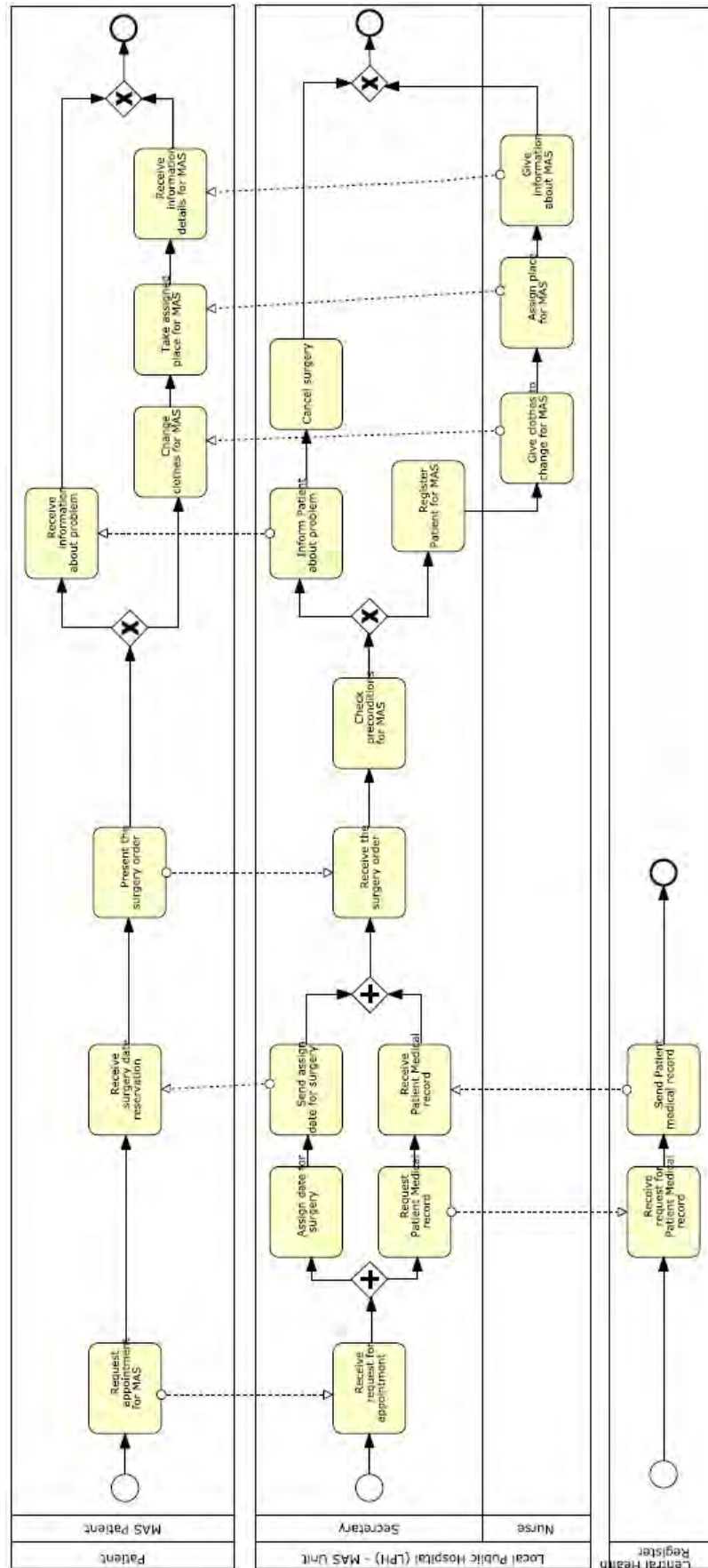


Figure 6.8.: “Patient Admission and Registration for Major Ambulatory Surgery (MAS)” in BPMN2

The calculation of execution measures for the BP case shown in Figure 6.9 then is as follows: first of all, the three base measures defined are shown for each activity, for example for the activity Assign date for Surgery these are:

- M1 (base) Enabled time of an Activity (ET) = 10-01-2010: 10:00
- M2 (base) Start time of an Activity (ST) = 10-01-2010: 10:30
- M3 (base) Completion time of an Activity (CT) = 10-01-2010: 10:40

Secondly, with these times we can calculate the rest of the measures, in the first place for the activity level, the working, waiting and total times for each activity which are shown in Table 6.14 using the formulae:

- M4 (derived) Working time of an Activity (AWoT = CT - ST)
- M5 (derived) Waiting time of an Activity (AWaT = ST - ET)
- M6 (derived) Total time of an Activity (ATT = AWoT + AWaT)

**Table 6.14.:** Calculation of execution measures at the activity level

Activity	Working Time (mins)	Waiting Time (mins)	Total Time (mins)
Receive request for appointment	10:00 - 09:30 = 30	09:30 - 09:30 = 0	30 + 0 = 30
Assign date for surgery	10:40 - 10:30 = 10	10:30 - 10:00 = 30	10 + 30 = 40
Send assigned date for surgery	10:50 - 10:45 = 05	10:45 - 10:40 = 05	05 + 05 = 10
Request Patient Medical record	10:05 - 10:00 = 05	10:00 - 10:00 = 0	05 + 0 = 05
Receive Patient Medical record	10:25 - 10:20 = 05	10:20 - 10:10 = 10	05 + 10 = 15
Receive the surgery order	11:05 - 11:00 = 05	11:00 - 10:50 = 10	05 + 10 = 15
Check preconditions for MAS	11:20 - 11:12 = 08	11:12 - 11:10 = 02	08 + 02 = 10
Register Patient for MAS	11:35 - 11:30 = 05	11:30 - 11:20 = 10	05 + 10 = 15
Give clothes to change for MAS	11:40 - 11:35 = 05	11:35 - 11:35 = 0	05 + 0 = 05
Assign place for MAS	11:55 - 11:45 = 10	11:45 - 11:40 = 05	10 + 05 = 15
Give information for MAS	12:10 - 12:00 = 10	12:00 - 12:00 = 0	10 + 0 = 10

After the times for the execution of all the activities in the BP case are calculated, the times for the complete BP case can be calculated as shown in the following using the formulae:

- M7 (derived) Total Working time of a BP case ( $TW_oT = \sum (AW_oT)$ )  
 $- = 30+10+05+05+05+05+08+05+05+10+10=98$  mins
- M8 (derived) Total Waiting time of a BP case ( $TW_aT = \sum (AW_aT)$ )  
 $- = 0+30+05+0+10+10+02+10+0+05+0=72$  mins

As there is a parallel branch in the BP, to calculate which executed path have to be taken into account to add up for the BP case times, first of all the calculation for the branches has to be performed taking into account the ATT for each activity in each path:

- Assign date for Surgery + Send assigned date for Surgery = 40+10 = 50 mins
- Request Patient Medical record+ Receive Patient Medical record= 05+15 = 20 mins

so the path that will be added up for the calculation of the TT for the BP case is the first one, leading to the following calculation:

- M9 (derived) Throughput Time of a BP case ( $BPTT = \sum (ATT)$ )  
 $- = 30+40+10+15+10+15+05+15+10=150$  mins.

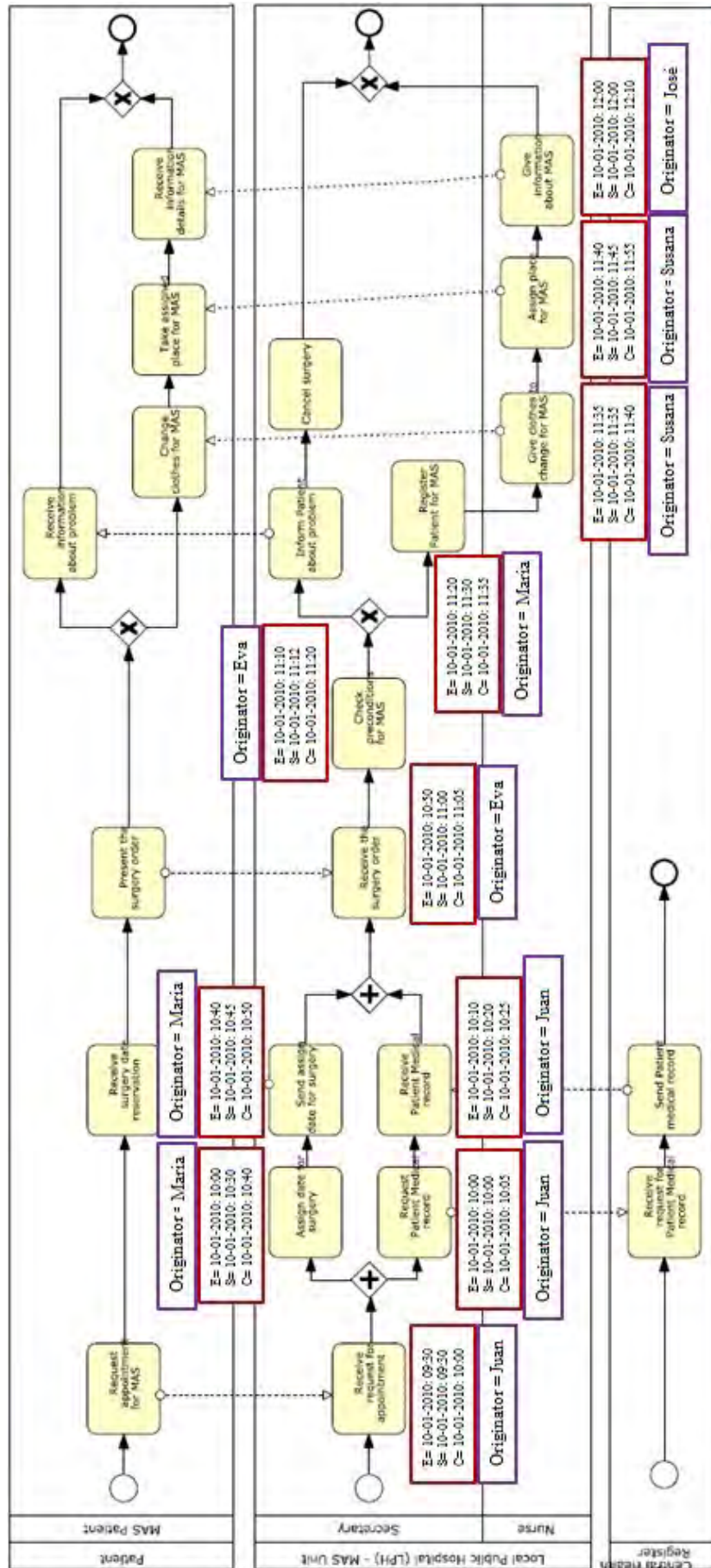


Figure 6.9.: Example BP case for the Patient MAS BP with execution times

After completing the calculations for the activity and BP case levels, we can calculate the rest of the measures for the BP level, defined as indicators. We will not show these here since they are mainly calculations of percentages for BP cases, and averages for all BP cases; it makes no sense to make values up to show these calculations. Instead we present the calculation of the index for activities and BP cases which provide information on the relation between working and waiting times, as follows:

- M10 (indicator) Activity Working time vs. Activity Waiting time index ( $ATI = AWaT/AWoT$ )  
Decision criteria = Index DC, for the activity Assign date for Surgery =  $30 / 10 = 3$  which will correspond to the High rank in the DC.
- M11 (indicator) Total BP Working time vs. Total BP Waiting time index ( $TTI = TWaT/TWoT$ )  
Decision criteria = Index DC,  $72 / 98 = 0.75$  which will correspond to the Medium rank in the DC.

### Time dimension, Capacity

For the calculation of the execution measures for Capacity as presented in section 6.3 the data shown in Figure 6.10 is used, where the marks P1 and X1 show a parallel gateway and a XOR gateway, respectively. The number of BP cases traversing each of the branches is shown in the red circles: for P1 both branches are always traversed which means all the BP cases execute all the activities in both branches in the execution event log, and for X1 the upper branch is traversed by only 30% of the BP cases (Patients) for which the Surgery is actually cancelled, and the lower branch is traversed by the 70% of the BP cases corresponding to the Surgeries actually performed. Other information needed is specified in the context data such as the number of resources assigned to each role in the BP model.

In Table 6.15 the data for the calculation of the execution measures in the BP Generic execution view, time dimension, capacity is shown, for an arbitrary execution of 100 BP cases, with columns: Activity name, Average activity working time for the execution of each activity (AATWoT), Role (Resource type) assigned for the execution of each activity which is taken from the BPMN2 model, and the calculation of the numbers of BP cases traversing each of the branches which corresponds to the number of BP cases processed by each activity as calculated by:

- M2 (base) Number of executions of each activity in all BP cases (  $NEA =$  count the times the activity is executed in all executions of BP cases)
- M3 (derived) Number of BP cases processed by each activity in the BP ( $NJA = NEA/Total$  BP cases)

**Table 6.15.:** Data from execution event logs already processed

Activity	Average AATWoT (min.)	Role = RABP	NJA = BP cases processed by each activity
Receive request for appointment	3	Secretary	$100/100 = 1$
Assign date for surgery	4	Secretary	$100/100 = 1$
Send assigned date for surgery	2	Secretary	$100/100 = 1$
Request Patient Medical record	3	Secretary	$100/100 = 1$
Receive Patient Medical record	2	Secretary	$100/100 = 1$
Receive the surgery order	4	Secretary	$100/100 = 1$
Check preconditions for MAS	6	Secretary	$100/100 = 1$
Register Patient for MAS	5	Nurse	$70/100 = 0.7$
Give clothes to change for MAS	3	Nurse	$70/100 = 0.7$
Assign place for MAS	5	Nurse	$70/100 = 0.7$
Give information for MAS	5	Nurse	$70/100 = 0.7$
Inform patient about problem	5	Secretary	$30/100 = 0.3$
Cancel Surgery	2	Secretary	$30/100 = 0.3$

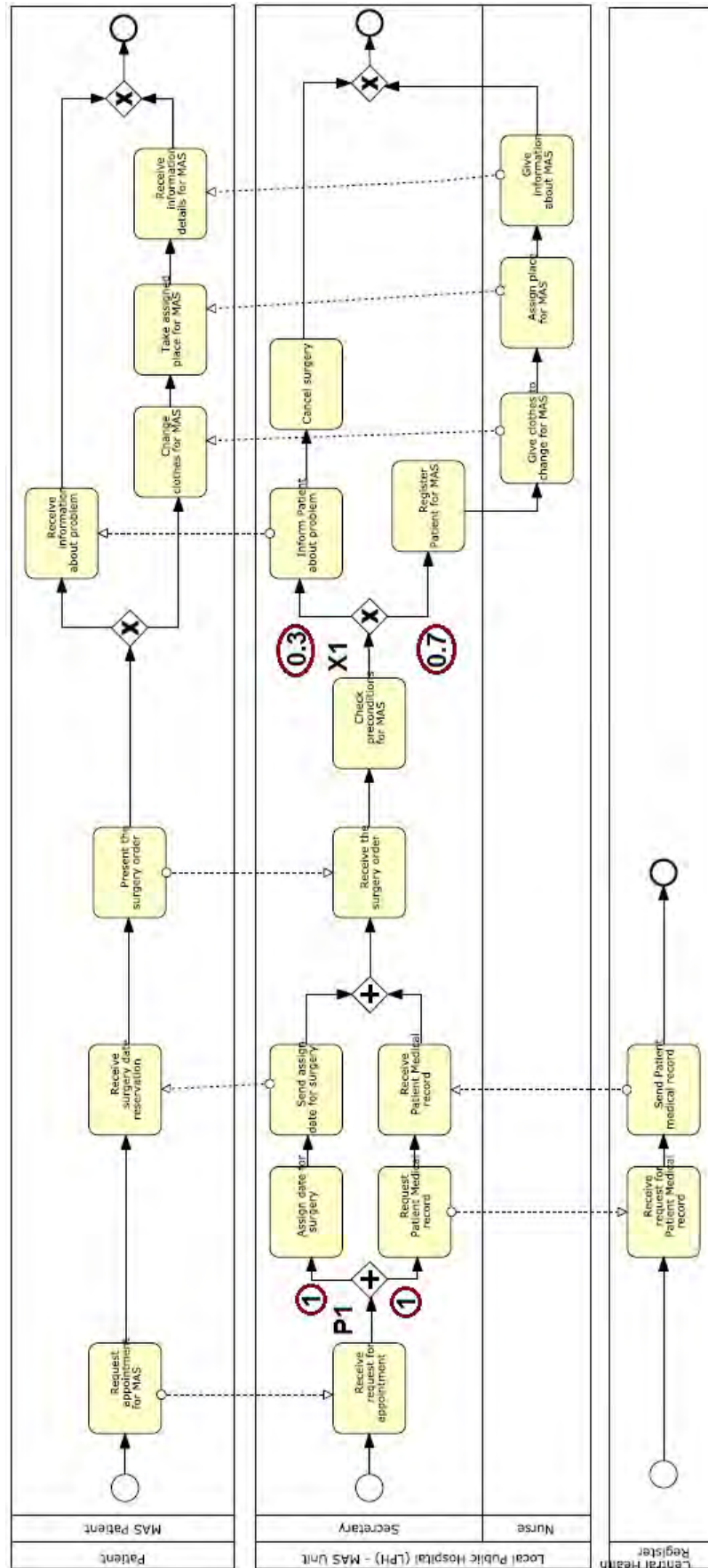


Figure 6.10.: Example BP case for the Patient MAS BP with BP cases for branch

Assuming the configuration of resources for each role defined in the BP model shown in the previous section, and that the number of resources assigned for each Role is defined in the context data as:

- Resource per role defined in the BP:
  - Secretary: Juan, María, Eva
  - Nurse: Susana, José

the rest of the measures are calculated as follows:

- M1 (base) Number of resources per role defined in the BP = NRRBP
- M4 (derived) Unit load for a resource in the BP ( $ULR = \sum (AATWoT * NJA)$ )
- M5 (derived) Unit capacity for each resource ( $UCR = 1/ULR$  indicates the number of jobs each resource can complete per unit of time)
- M6 (derived) Pool capacity for each role in the BP ( $PCR = UCR * NRRBP$ )
- M7 (derived) Process capacity of the BP ( $PCBP = \text{Bottleneck of the BP} = \text{smallest of measure 6}$ )

the table for calculation of these measures is shown in Table 6.16 with columns: Role (Resource type), Unit load, Unit capacity, Available resources and Pool capacity.

**Table 6.16.:** Process capacity calculation for the example Patient MAS BP

Role	Unit load (min.) = ULR (M4)	Unit capacity = UCR (M5) (BP cases/min.)	Available resources = NRRBP (M1)	Pool capacity = PCR (M6) (BP cases/min.)
Secretary	$3 \times 1 + 4 \times 1 + 2 \times 1 + 3 \times 1 + 2 \times 1 + 4 \times 1 + 6 \times 1 + 5 \times 0.3 + 2 \times 0.3 = 26.1$	1/26.1	3	$3/26.1 = 0.12$
Nurse	$5 \times 0.7 + 3 \times 0.7 + 5 \times 0.7 + 5 \times 0.7 = 12.6$	1/12.6	2	$2/12.6 = 0.16$

Based on the calculations presented, the Secretary role is able to handle 0.12 BP cases per minute or 7.2 BP cases per hour, and the Nurse role is able to handle 0.16 BP cases per minute or 9.6 BP cases per hour; the process capacity (M7) is therefore the same as the smallest pool capacity of the process which is Secretary, corresponding to the bottleneck of the process.

### Cost dimension

For the cost dimension several execution measures calculated previously are used, and information from the context data as the cost per resource:

- M1 (base) Resource cost per unit of time = RCT (from context data), for the example in euros:
  - Juan = 25 per hour; María = 20 per hour; Eva = 20 per hour; Susana = 15 per hour; José = 10 per hour

which allow the calculation of the measures for the operational cost of the BP based on human resources, presented for the BP case of the example in Table 6.17:

- M2 (derived) Cost per activity in a BP case ( $ACo = AWoT * RCT$ )
- M4 (derived) Total cost of BP case ( $TCo = \sum ACo_{(i)}$ )

The calculation of the execution measures presented is very simple and it is given only to illustrate them, the rest of the measures for this dimension involve the calculation of percentages and averages for all BP cases for the indicators; it makes no sense to make values up to show these calculations, so they are not shown.

**Table 6.17.:** Cost measures for the example BP case

Activity	AWoT (min.)	Resource	RCT (euros)
Receive request for appointment	30	Juan	30x12.5=375
Assign date for surgery	10	María	10x3.3=33
Send assigned date for surgery	05	María	05x1.66=8.3
Request Patient Medical record	05	Juan	05x2.08=10.4
Receive Patient Medical record	05	Juan	05x2.08=10.4
Receive the surgery order	05	Eva	05x1.66=8.3
Check preconditions for MAS	08	Eva	08x2.66=21.28
Register Patient for MAS	05	María	05x1.66=8.3
Give clothes to change for MAS	05	Susana	05x1.25=6.25
Assign place for MAS	10	Susana	10x2.5=25
Give information for MAS	10	José	10x1.66=16.6
Total cost for the BP case = TCo	-	-	522.83

## 6.5. Conclusions

In this Chapter the BPEMM execution measurement model has been presented which provides a set of execution measures for the execution of BPs realized by services. BPEMM is part of the BPCIP continuous improvement process that is defined in the methodological dimension of MINERVA framework defining its complete lifecycle to guide the improvement efforts in organizations. By means of the GQM paradigm, the model aims to relate business and BP goals defined by the organization to the real execution of BPs, defining Goals, Questions and the execution measures for each defined Goal and Question.

Other elements have been integrated in the definition of BPEMM such as the three views for grouping the measures: BP generic execution, BP Lean execution and Services execution, which are organized by means of the “Devil’s quadrant” dimensions of time, cost, quality and flexibility, also defining a hierarchy with three levels of execution measures for activity instances, BP cases and the BP for all BP cases. These three dimensions (defining the BPEMM “cube”) are a systematic classification of execution measures which is itself an added value because it facilitates the use of BPEMM against the alternative of a simple flat list of measures.

Existing execution measures were integrated and new ones defined, which allow an organization to select, implement, collect and analyze measurement results for the improvement of its BPs. An example of the use of some of the BPEMM execution measures for the Patient MAS BP from the Hospital General de Ciudad Real (HGCR) was presented to illustrate the calculation of the execution measures defined.

Business people are an integral part of the whole improvement effort, which is also supported by the IT people to provide the technical support needed for the measurement activities and the implementation, collection and calculation of the execution measures from BPEMM.





## Chapter 7.

# Business Process Service Oriented Methodology (BPSOM)

This Chapter describes the Business Process Service Oriented Methodology (BPSOM) that has been defined in MINERVA framework to guide the service-oriented and model-driven development of business processes.

The Chapter is organized as follows: in section 7.1 a description of the BPSOM is presented, in section 7.2 the Disciplines and elements defined by BPSOM are presented along with the use of BPMN2 and SoaML to model BPs and services respectively, and in section 7.3 the Phases proposed to guide the use of BPSOM are described. In section 7.4 the implementation of BPSOM as an EPF Composer method plug-in is presented and finally in section 7.5 conclusions for the chapter are discussed.

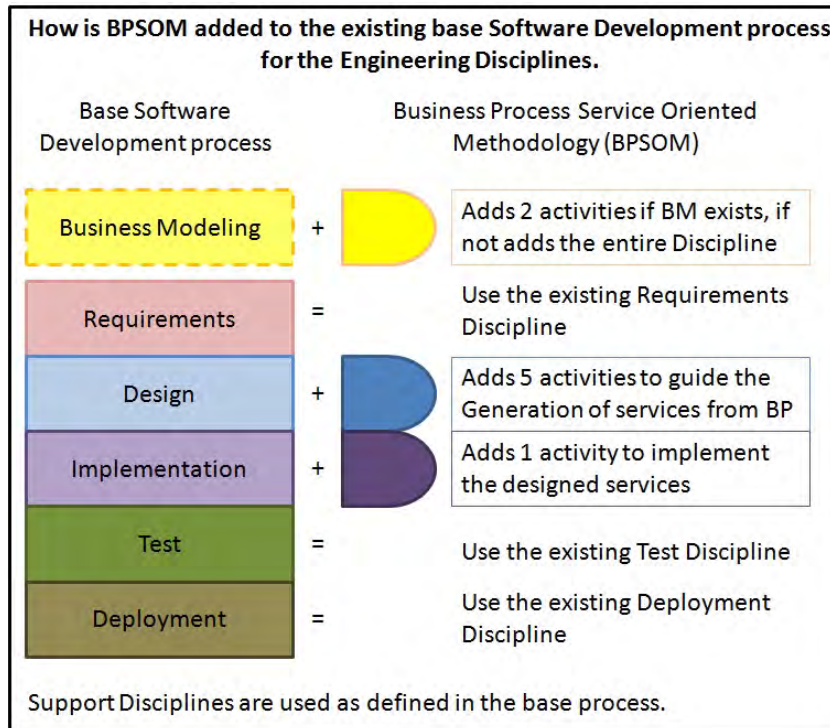
The contents of this chapter are complemented by the contents in the following chapters: chapter 5 which presents the Business Process Continuous Improvement Process (BPCIP) defining the extended BP lifecycle, and chapter 8 which describes the MDA approach included in BPSOM for generating SoaML service models from BPMN2 models.

### 7.1. Introduction

The Business Process Service Oriented Methodology (BPSOM) is defined in the methodological dimension of MINERVA, and its main objective is to guide the development of BPs by means of services based on models. BPSOM has been defined to be integrated into the existing base software development process used in the organization. Our aim is to reuse existing knowledge in the organization, adding only specific activities to guide the service oriented development from BPs.

An initial definition of the methodology was carried out as a master's thesis [Delgado, 2007] as an extension and adaptation of the Unified Process [Jacobson et al., 1999] using the Rational Unified Process [RUP-IBM, 1999-2011] implementation. This initial definition was then abstracted from the base software development process (by such means as eliminating references to RUP artifacts, roles and activities and by adjusting the definition of elements) and integrated into MINERVA framework [Delgado et al., 2009b] as an extension of any base software development process.

The BPSOM version presented here corresponds to its extension [Delgado et al., 2011c] by adding service modeling in SoaML to the Design Discipline along with the automatic generation of SoaML service models from BPMN2 BP models through QVT transformations that will be described in chapter 8. Figure 7.1 shows the general definition of BPSOM and its use with the existing software process.



**Figure 7.1.:** How BPSOM is added to the existing software development process

The Disciplines that BPSOM defines as being the main ones for service-oriented development from BPs are: Business Modeling, Design and Implementation, to identify and model BPs, identify and derive the services making up these processes, as well as to design and build them, respectively. Each Discipline includes the definition of the key activities needed to accomplish service definition, specification, design and implementation, in addition to roles involved, artifacts to be produced and templates for them.

Other necessary Engineering Disciplines such as Requirements, Testing and Deployment are taken from the existing software development process in the organization, in order to follow the process that users are accustomed to. Support Disciplines such as Quality Assurance, Software Configuration and Project Management are used as defined in the existing base process. As it is shown in Figure 7.1 in order to add BPSOM to the existing base software development process in the organization, the defined Disciplines and their elements are added to those that already exist.

From Requirements to Deployment the so called Engineering Disciplines are assumed to exist in the process, and the activities and other defined elements are therefore added to them. If the Business Modeling Discipline exists, the activities and other defined elements are added to it, if not, the entire Discipline is added to the process in order to get better insight of the organization and their BPs.

As mentioned in chapter 5 BP modeling is of great importance to the organization for several reasons; it is a key factor in showing explicitly how the business works and in being able to reason about business based on the models. From the point of view of software development, Business Modeling and Requirements Disciplines are complementary and are carried out at the same time, since the second one, based as it is on the first, aims to specify the characteristics of the software needed to support the business operation defined, for example what the user forms should look like, as well as the data to be displayed and used throughout the BP execution. The flow and relationships between the activities are shown in Figure 7.2 as a BPMN2 model.

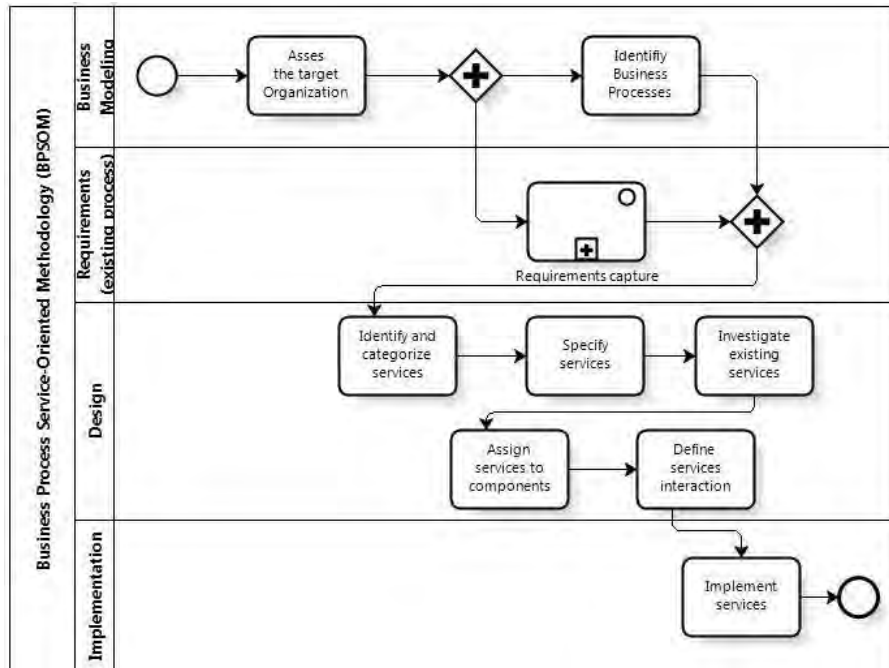


Figure 7.2.: BPSOM activities flow as a BPMN2 model

BPSOM is part of the complete lifecycle defined by MINERVA for the continuous improvement of business BPs on model-driven service-oriented development and execution measures, as presented in chapter 5. In Figure 7.3 the method of work of MINERVA framework as presented in chapter 4 is shown, to set the reader in the context of the definitions that are described below. The use of BPSOM is shown in the rectangle between BPMN2 models and SoaML service models, and includes the transformations for the automatic generation of services.

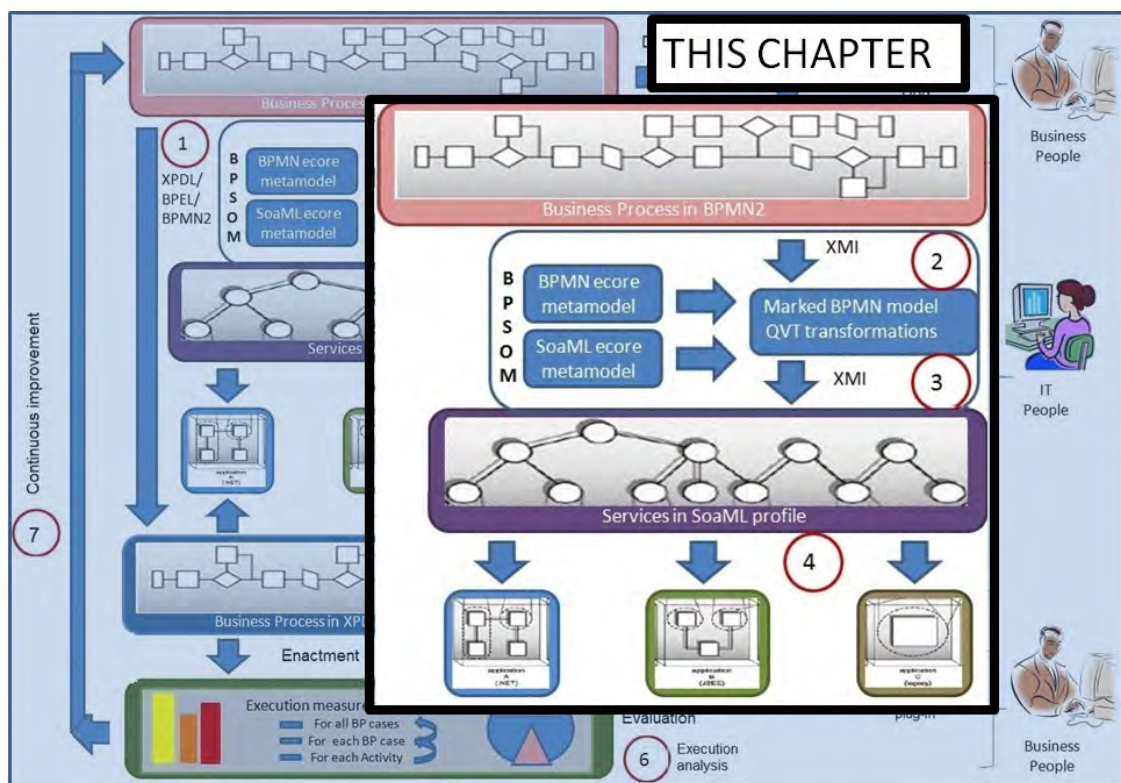


Figure 7.3.: MINERVA method of work through BPCIP

BPSOM is used in the BPCIP lifecycle to navigate from the Design&Analysis phase to the Configuration phase, as a guide for developing service-oriented systems from BPs by means of carrying out the activities defined. A reference BPSOM lifecycle is proposed which also defines phases, but this is for guiding the software development process. In the BP lifecycle [Weske, 2007] only the Configuration phase deals with the development of systems to support BPs but without specifying a methodology or process to carry that out, only an informal guide.

As BPCIP, the BPSOM methodology has also been implemented as an EPF Composer [Eclipse, 2004-2011] Method plug-in [Delgado et al., 2010a] in order to provide interoperability with other processes defined in the same way, and published it as a Web site [Delgado, 2010-2011] to be easily accessed and used.

BPSOM is described in the following sections based on two different views: Disciplines and Phases, where the first one defines the Disciplines with activities and elements needed for service-oriented development, and the second presents the proposed reference BPSOM lifecycle for guiding the service-oriented development in an iterative and incremental way.

## 7.2. BPSOM Disciplines

The Disciplines of BPSOM define the activities, artifacts and roles needed to guide the service-oriented development from business processes. The Disciplines and their activities -including recommended methods and/or techniques-, input and output artifacts and roles they comprise are presented below.

### 7.2.1. Business Modeling Discipline

The purpose of the Business Modeling Discipline is to ensure that developers and other stakeholders have a common understanding of the Organization and to derive the requirements for the software service system, linking them to the BPs identified. The goal is to obtain a map of the organization and its BPs, from which to gain a better understanding of the business and requirements for the services and software system. To achieve this, several meetings with business people in the organization take place. When BPSOM is used in BPCIP the activities are performed as defined in BPCIP in a similar way.

As mentioned in chapter 5 we have defined that the notation used to specify BP models is BPMN2. To take full advantage of BPMN2 benefits, the BP modeling activity is carried out by business people alongside with the software team carrying out the development effort. Nevertheless, business people can also model BPs in BPMN2 by themselves after the meetings, and can then give them to the software area for the services development to be undertaken. All modeling information can be stated in the BP model using a BP Modeler tool providing BPMN2 support (such as BizAgi, Oryx, Activiti), although a requirement for using the defined QVT transformations is that the tool should provide facilities to save the model in BPMN2 (XML) serialized format.

The roles involved are Business Analyst and IT Analyst and Architect (from here on referred to as Analyst and Architect, respectively), who work in conjunction to identify, model and validate the BPs of the organization. The main deliverables are the Assessment of the target Organization which includes the key aspects identified, together with the BP document which contains the models specified in BPMN2 and any other useful information about them that is to be used in the development. The BPMN2 models are also attached in BPMN2 format (XML) to be loaded in the development environment and they are also registered in a central BP Model repository, to provide easy access to them. In Figure 5.2 the activity detail diagram for the Business Modeling Discipline is shown, with the role responsible corresponding to the IT Analyst.

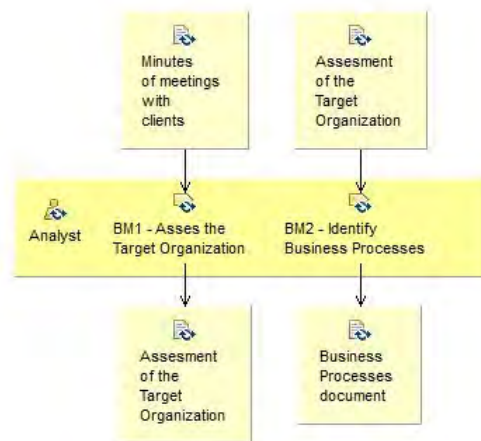


Figure 7.4.: Business Modeling activity detailed diagram

#### 7.2.1.1. BM1 - Assess the target Organization

This activity aims to engage the project team with the organization for which the development is being carried out. It may be the case that the software area belongs to the organization then, documents may therefore exist that provide the needed insight, among other things, into the area of business, business goals, operation, employees, technologies used in the organization. Where that occur these documents can be used rather than their being produced again in the meetings with business people. The OMG Business Motivation Model (BMM) OMG [2010] can also be used to express the goals, strategy and the organization's other business information, which may then be linked to the definition of SoAML services.

**Inputs** Information giving insight into the organizational context, provided by business people in the course of the meetings carried out in conjunction with the Analyst and Architect.

**Outputs** Assessment of the Target Organization document, in which the information gathered is structured in sections to provide a clear view of the organization's current situation. This document will be discussed with business people until all stakeholders agree upon it.

**Roles** The role responsible for this activity is the Analyst, the Business Analysts and the Architect also participate, providing their business and technical knowledge, respectively.

**Method** It is important to know about the capabilities of the organization in which the BPs are to be modeled, implemented, deployed and executed. This includes available technologies and infrastructure, sections and people involved, business goals, required and available tools, as well as clients and partners of the organization, problems and improvement opportunities. To carry out the meetings with business people a questionnaire can be drawn up, which would ask for the information needed, and a checklist on the questionnaire sections can also be developed, to be sure that all the issues have been discussed. Existing techniques for conducting meetings and guides for checklists can be used [IEEE, 2004].

#### 7.2.1.2. BM2 - Identify Business Processes

This is one of the key activities in the development of services from BPs, since it is the main input needed to understand and describe BPs in the organization, as these are mainly those related to the application that is being developed. The notation used to specify the BP models from which

to generate SoaML services models is BPMN2. Other notations could be used, but the QVT transformations defined could not then be executed.

All the elements in the BPs have to be identified and modeled, such as participants, roles, activities performed and their logical sequence, as well as message flows between participants, diverging and converging paths. Several aspects have to be taken into account when modeling BPs such as process patterns (workflow patterns) [van der Aalst et al., 2003a], BP modeling guides [Mendling et al., 2010] and specifying business rules in BPs [OMG, 2008d].

**Inputs** Assessment of the Target Organization document which includes the information from the meetings with business people about the BPs performed in the organization.

**Outputs** The Business Processes document in which BPs are identified and modeled including all the elements needed to specify them, and the BPs in BPMN2 (XML) format.

**Roles** The role responsible for this activity is the Analyst, the Business Analysts and Architect also participate, providing their business and technical knowledge, respectively.

**Method** To identify and model BPs in the organization several meetings with business people have to be carried out, for which existing techniques and checklists [IEEE, 2004] can be used, such as Facilitated meetings (in SWEBOK carried out in the Requirements Discipline). This aims to get people together in order to gain better insight into the organization and their BPs and requirements, instead of interviewing them individually. One advantage of this technique is that conflicts between different views and ways of performing work in the organization will surface early in the development process and business people can thus recognize and solve them, in their quest to define clearly the activities, data and responsibilities to be specified in the BP models. It should be said, however, that meetings need to be handled carefully, designating a moderator to prevent conflicts for growing.

The Business Analyst role we have defined plays a key role in this activity, as business people have the knowledge of the organization's business to model the BPs correctly. The use of process patterns [van der Aalst et al., 2003a] helps in various BP modeling issues, since they provide a great insight into common modeling problems along with solutions these. Collaborative BPs modeling is of key importance since different interacting organizations are involved; they need to agree on their interaction points if the collaboration is to take place.

When it is not possible to have meetings with people from other organizations involved, the interaction points can be defined in a conceptual manner (i.e. stating what should be carried out by the message interaction at each interaction point) and the specific definition of the interaction can be further investigated when the services are designed. The BPs document includes the BP models and when needed, descriptions in natural language can also be provided to give a better insight into specific parts of the graphical model, or to describe the flow of the complete BP as an introduction to the model.

### **BP modeling with BPMN2**

As mentioned before the modeling of BP in MINERVA framework is done using the BPMN2 notation, which provides most of the elements needed for specifying BP models. Figure 7.5 shows the "Patient Admission and Registration for Major Ambulatory Surgery (MAS)" BP from the Hospital General de Ciudad Real (HGCR) simplified and adapted to be used as example, as presented in chapter 6.

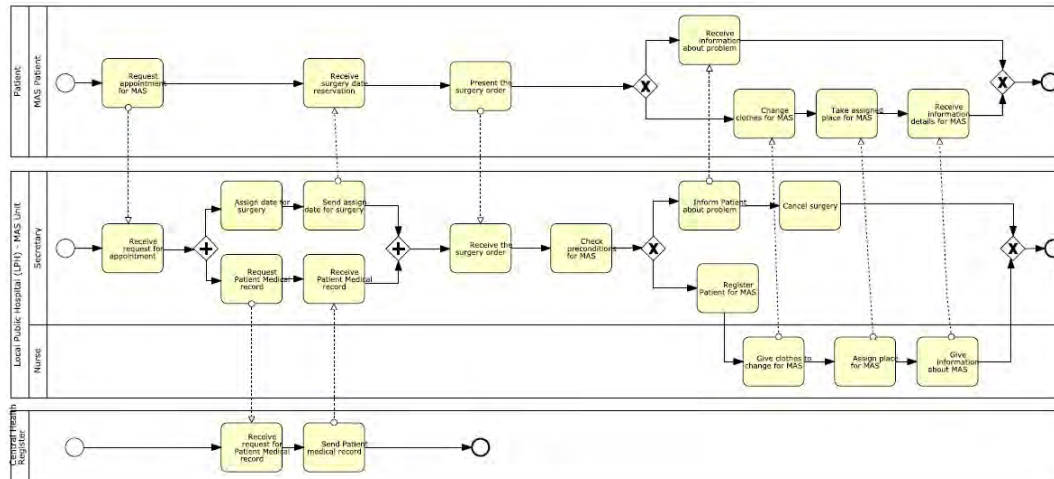


Figure 7.5.: Patient MAS business process in BPMN2

As can be seen in Figure 7.5 several message flows are defined between the participants in the collaborative BP, but not all of these are to be automated; the ones that are manual steps of interaction between the nurse and the patient are examples of this. These interactions show physical interaction between the participants when the patient is at the hospital in order to undergo the MAS. Other messages flows can be automated or not automated, such is the case in the communication of the surgery date to the patient: it may be decided that the communication will be by means of an email, an sms or an automatic telephone call, or it may be that the option chosen is for the secretary to call the patient and give him or her the information. The interaction with the Central Health Register participant will be automated, however, since this involves exchange of data regarding the patient's medical record.

### 7.2.2. Design Discipline

The purpose of the Design Discipline is that of identifying and designing services to perform the defined BPs, as well as to specify their interfaces with their operations and parameters, the components that will implement them, and reusing existing services in the organization as much as possible. We have included specific steps for the automatic generation of SoAML models from BPMN2 models in the BPSOM activities, to help the development process; these are described in chapter 8. The defined activities can also be carried out manually, however, i.e. without applying the transformations we have defined, as the rules we have created are also a guide for the design of the services for realizing the BP.

As for BP modeling, there is also a great variety of notations that can be used for service-oriented modeling [Delgado et al., 2010g], with UML being the one which is most widely-used. As SoAML is a UML profile and metamodel which extends UML by adding specific elements for service modeling and is also an OMG standard, as described in chapter 3, it is being rapidly adopted by the software community. Based on these and the fact that MINERVA is a standardized framework for construction, we have defined that the notation used to specify service-oriented models generated from BPMN2 models is SoAML.

To guide the modeling of services with SoAML, in each activity defined we have added which SoAML diagram has to be specified and how, as well as when a corresponding diagram to be used in the activity exists in the standard, and when to perform the activities manually. When applying the automatic generation of SoAML service models from BPMN2 models, each activity is used as a guide for the visualization of the corresponding SoAML diagrams, extending them as needed, with information that can not be automatically generated.

To be able to specify SoAML service models, using either automatic generation or a manual design based on the guides provided, we provide a distribution of MINERVA called *Eclipse MINERVA design*, which includes all the plug-ins needed to carry out the activities specified and the automatic

generation, including the Eclipse SoaML plug-in we have developed for SoaML modeling. The description of the tools support provided is given in chapter 9.

The roles involved are Architect, Analyst and Developer, with the Architect being responsible for the whole discipline, including the specification of the details for extending the BPMN2 model, so the corresponding services can be identified and generated, when using automatic generation. The main deliverables of the Design Discipline are the Services document, with the services identified and specified, and the central Services Catalogue, to include new generated services and to search for existing ones in the organization, to reuse the latter. The Services document includes the SoaML service-oriented models, which are also attached in digital format (XMI). In Figure 7.6 the activity detail diagram for the Design Discipline is shown, with the associated responsible role defined, corresponding to the Architect.

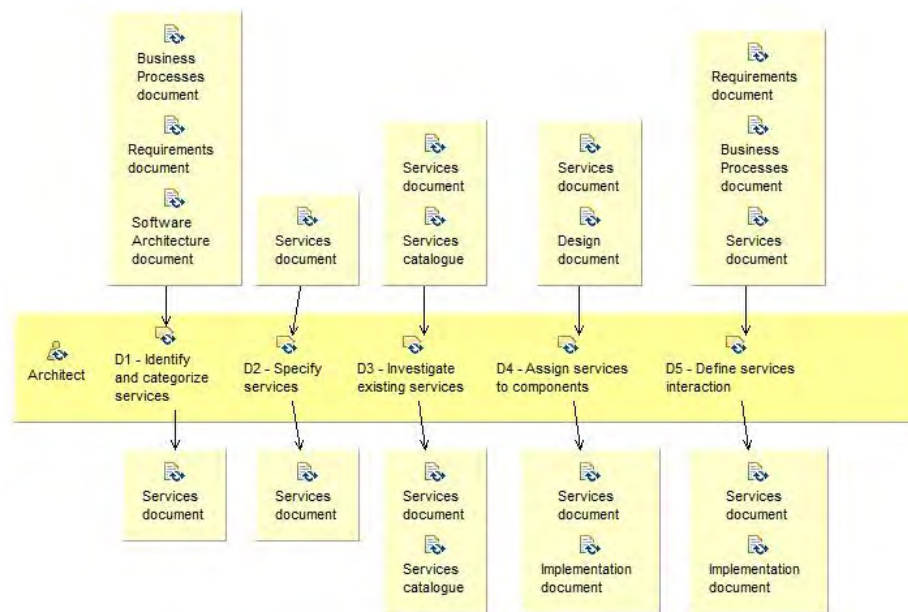


Figure 7.6.: Design Discipline activity detailed diagram

### 7.2.2.1. D1 - Identify and categorize services

This activity is defined to identify the services needed to realize the BPs under development and is a key one in the approach. Services that the organization needs to provide to other parties along with services that the organization needs to consume from other parties are identified based on the messages that they exchange, and each party is defined by a pool in the BPMN2 model.

Services which are internal to the organization are identified within the lanes each pool presents, based on the identification of activities that can be carried out in an automatic way. Both types of services have to be identified by the Architect based on his or her knowledge of the BP in the organization and the interactions with external participants, gained by means of the meetings carried out with the business area.

The categorization of services helps define a hierarchy that allows the services and their dependencies to be organized. The categories to be used are determined in the organization by means of selecting from among existing ones, such as Utility services, Business services and Coordination or Process services as defined in [Erl, 2005], or Basic services, Intermediary services, Process centric services and Public Enterprise services as defined in [Krafzig et al., 2005], or the organization may create its own categorization.

**Inputs** BPMN2 models specified in the BM2 activity, and the Software Architecture Document (SAD) defined in the existing base software development process. providing a general view of the system.



**Outputs** The Services document in which the services are identified and categorized, with the corresponding Identify and categorize services section informed.

**Roles** The role responsible for this activity is the Architect, but Developers and Analysts also participate to provide their technical and domain (business) knowledge, respectively.

**Method** The Participants in the SoaML service-oriented diagrams are identified by means of the existing Pools in the BP (definition of Pool in BPMN2), so, for each Pool we identify a general Participant for the Services Architecture. To identify the services for supporting the BP realization, each exchanged message between pools is analyzed, to see whether this flow can be automated or not, depending on the participants and the semantics of the interaction. When the activities that will be supported by services are established, its type is set to “Service” to show this intention in the BPMN2 extended model. This identification is carried out from the point of view of the provider of the service.

When considering the direction of the messages exchanged, we therefore define that the activity to which the message is an incoming message is the one to be supported by a service. The activity from which the message is an outgoing message is therefore the consumer of the service. The pool containing each activity will define the participant that provides or consumes the service, depending on where the activity is supported or where it invokes the given service. This information will be then set in the Service Contract for the service defined, indicating the consumer and provider roles as established.

After defining the services for the collaborative BP in which the given participants interact, each general participant can define its own internal services to realize internal activities that will accomplish the defined interactions with external participants, as well as supporting the internal realization of the BP. For doing this, internal participants corresponding with the lanes in the associated pool are further defined, and the internal Services Architecture is also specified, by identifying the activities that will be supported by services. This identification is guided by the Architect’s knowledge of the BP and its operation in the organization. No generation has been provided for internal services as yet, and this line of work is left for future research.

The Software Architecture document from the base software development will include as one of its views the services view defined in the Services document, which contains the service models to support the BPs, as defined. Other technical information will also be specified in the Software Architecture document that will help to organize the entire system under development, and placing the services correctly in the complete picture, for example relating them with the BP engine that will execute the BPs. The Services document’s focus is on the definition and design of services in the corresponding models.

### **Service-oriented modeling with SoaML**

The use of SoaML in this activity implies specifying the Services Architecture (SA) diagram which defines the participants, the contracts for the services defining their interaction, and the roles they play in each one as provider or consumer of the service. At this stage of the development the definition of services is carried out at a high level which allows us to show the “big picture” of the services needed, along with which particular participants they connect.

Figure 7.7 presents the SoaML ServicesArchitecture diagram corresponding to the example. Each Participant corresponding to each pool in the BP is shown: “generalHospital”, “patient” and “centralHealth”, along with the ServiceContract for each identified service. Each Participant has a role in the collaboration that may be as provider or consumer, which is also shown in the diagram. This connection also defines the Ports (Service,Request) in which each Participant provides or consumes services.

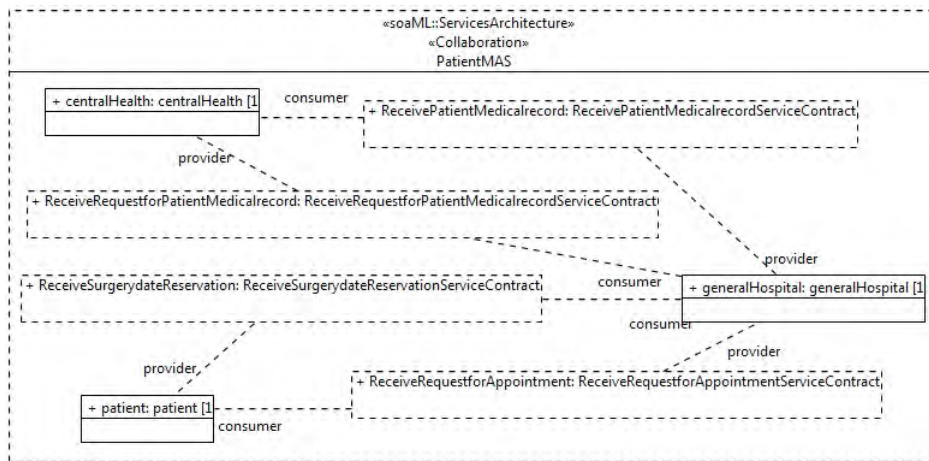


Figure 7.7.: SoaML ServicesArchitecture diagram for the PatientMAS BP

Although at this point of the development, services are only sketched and not completely specified when performing the activities in a manual way, the Participants diagram can be also modeled as the ports in which the services are provided and consumed are defined in the Services Architecture diagram. In this case, the Service and Request Ports for the identified services can not yet be typed since the corresponding interfaces are not specified. Figure 7.8 shows the Services and Request Ports for the defined services for each participant identified in the BP example.

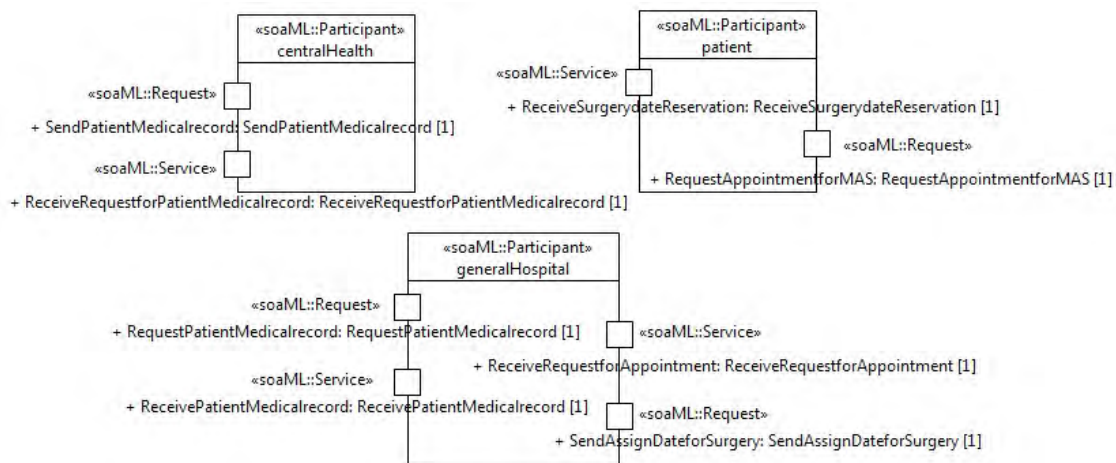


Figure 7.8.: SoaML Participants and Services diagram with Ports

The Participant “generalHospital” provides the services “ReceiveRequestforAppointment” and “ReceivePatientMedicalrecord” as indicated by the Service type of the Port. In the first case, it provides the “ReceiveRequestforAppointment.” interface and requires the “RequestAppointmentforMAS” interface from the consumer, i.e. the patient, who consumes the service as indicated by the Request type of the associated Port. It provides the “RequestAppointmentforMAS” interface as required by the “generalHospital” to answer the request for the Appointment, and consumes the “ReceiveRequestforAppointment” interface provided by the “generalHospital”, to request the Appointment.

The design option presented is due to the definition of the service being bidirectional, thus defining the need for one interface in each participant to carry out the interaction. The rest of the services provided and consumed for all the participants defined, are modeled in the same way using the bidirectional communication design from the SoaML standard. The other options for the services design defined in the SoaML standard -unidirectional and bidirectional with ServiceInterface- will be presented in chapter 8, along with the transformations defined.

#### 7.2.2.2. D2 - Specify services

The specification of services corresponds to the definition of all the information needed to further implement them. This includes all the information for the Service Contract: interfaces provided by the service, operations provided by the interfaces, input and output parameters of each defined operation including name and type, provider and consumer roles defined in the service contract for each interface, and pre and post conditions for the service execution, when needed.

The service contract specification allows to reason about the defined services, providing the basis for modifying them prior to their implementation. The specification of services by means of interfaces enables the definition of the functionalities that are provided by the service to be separated from its actual implementation, which is defined in a specific activity that is carried out after the service has been completely specified.

**Inputs** The Services document which has the information regarding the services identified in the Identify and categorize section.

**Outputs** The Services document with the information added for the specification of services carried out in the Specify services section.

**Roles** The role responsible for this activity is the Architect, but the Analyst also participate to help in the specification of the services as it is based on the BPMN2 model.

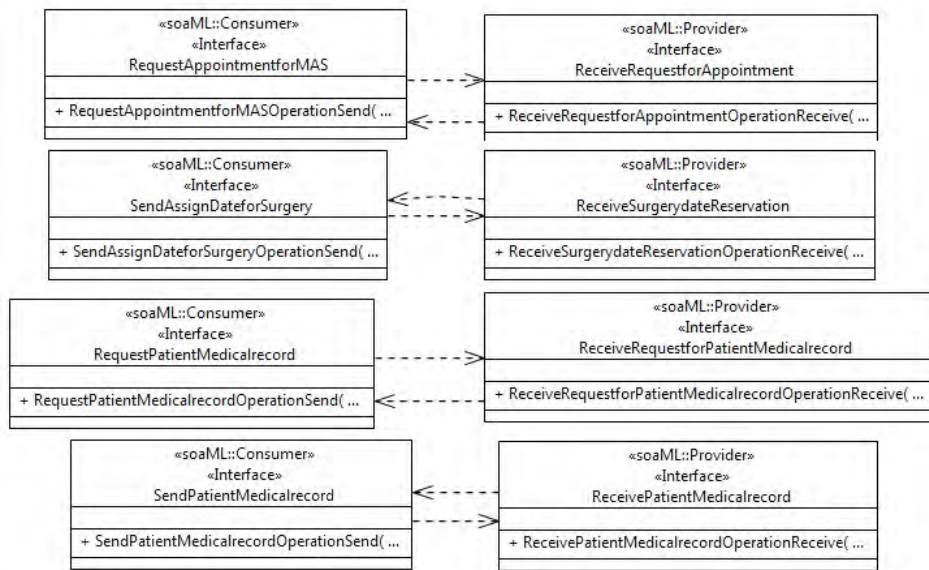
**Method** After services are identified and modeled, design details must be specified to be able to implement them further. The interfaces that the service provides and consumes have to be defined, along with the operations, parameters and corresponding type. All elements needed have to be modeled including the relationships among provided and consumed interfaces, defining the roles of the interfaces in the Service Contract. The Service Contract defines how the service is used (invoked) as well as its associated behavior and response in normal and abnormal situations, which can be shown in a choreography attached to the ServiceContract, by means of a UML sequence diagram.

At least one operation has to be provided by each defined interface, containing at least one input parameter (for sending the data to issue the request) and one output parameter (which if not needed can be used to return and acknowledge of the invocation). The operation provided will offer the functionality identified for the service, and if more functionality is required other operations can be added to the interface.

As the interactions are supported by the services between different organizations (pools in the BPMN2 model) special care has to be taken when defining the signature of the operations in the interface, as modifications both in parameters and their types can be a difficult task once the BP is executing. When the interaction involves more than one operation invocation, the specification of the choreography for the service gains more importance in showing the sequence.

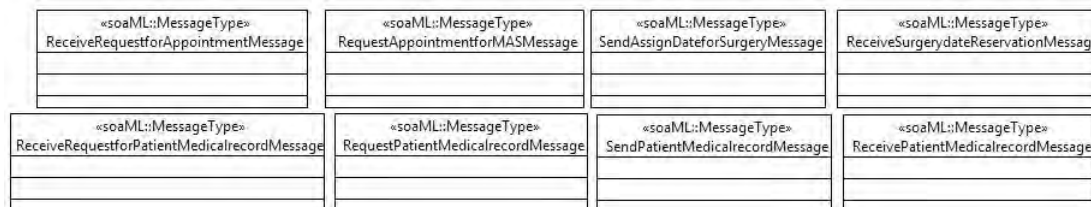
#### **Service-oriented modeling with SoaML**

The use of SoaML in this activity implies specifying the service Interfaces (SI) diagram, the Service Contract (SC) diagram and the Messages Type (MT) diagram. In each type of diagram the defined data is modeled by means of the guides provided. When the automatic generation is used, these three diagrams are also generated from the identification of services carried out in the previous activity, so in this activity the diagrams are visualized and the generated elements are extended with data that can not be generated, for example the attributes for the Messages Type. The specification of services Interfaces is shown in Figure 7.9 for the services defined for the example BP, using the bidirectional communication pattern with UML simple Interfaces.



**Figure 7.9.:** SoaML Interfaces diagram for the defined services

With the automatic generation of services, the parameters defined in the Interfaces operations are typed with the MessageTypes defined for the messages to be interchanged by the participants. As mentioned, the signature of the operations is of key importance for interactions between different participants, so we propose to define one MessageTypes to type the input and return parameters for each operation, so preventing problems when changes arise, as an XML Schema can later be defined for the internal structure of the message. This allows each participant to change only the encoding and decoding of the message while the signature of the operation remains unchanged. Figure 7.10 shows the definition of the MessageTypes to be used as parameters in operations, for the services defined for the example BP.



**Figure 7.10.:** SoaML MessageTypes diagram for the defined services

As can be seen in Figure 7.10 in the case of services generation the MessageTypes have no attributes specified as it is not possible to generate them from the BPMN2 model, since this kind of definition needs semantic knowledge of the functionality defined for the service. In this activity, whether the services are specified manually or generated automatically, the MessageTypes diagram has to be extended to include the attributes for each MessageTypes defined.

After the specification of the service Interfaces has been completed, the ServiceContract for the service can be specified. The consumer and provider roles are defined and typed with the defined interfaces, according to the consumer and provider service definition. Figure 7.11 shows the ServiceContracts for the services and corresponding interfaces defined for the example BP.

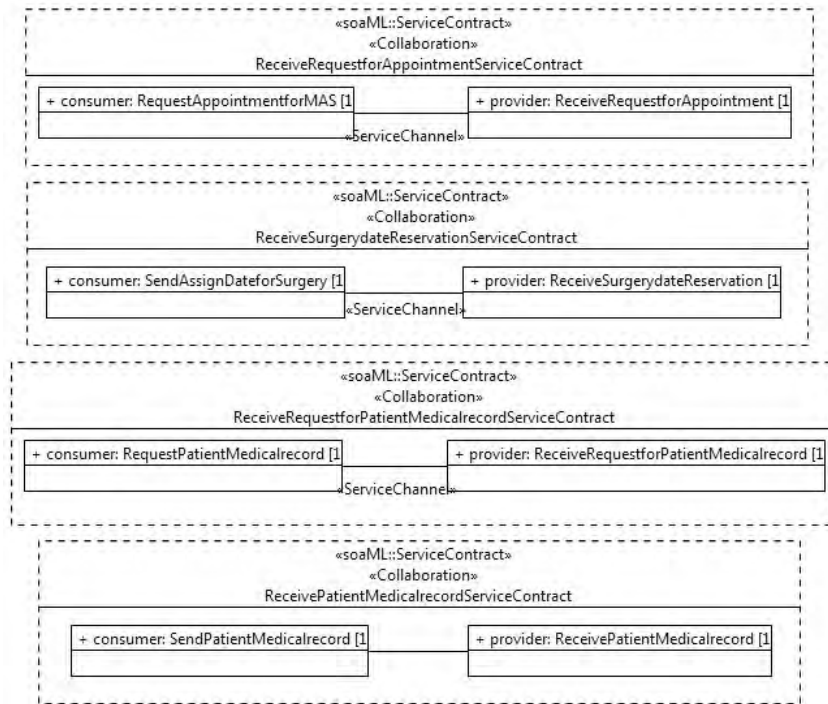


Figure 7.11.: SoaML ServiceContracts diagram for the defined services

A ServiceContract can have a choreography associated which can be any type of UML interaction diagram, an example of the choreography associated with the ServiceContract for the service ReceivePatientMedicalrecord as a UML sequence diagram is shown in Figure 7.12.

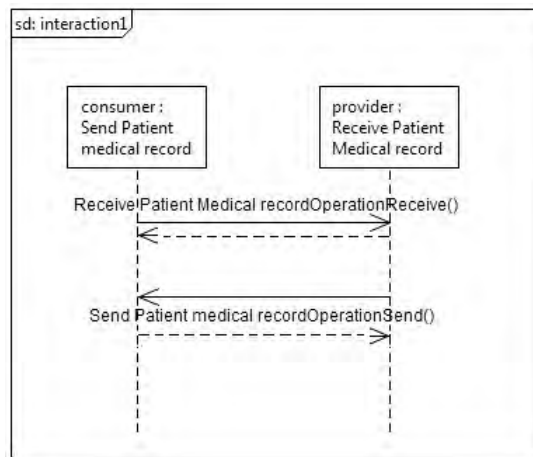


Figure 7.12.: SoaML choreography diagram associated with the ServiceContract

The invocation of the provider by the consumer is the initiating event for the interaction and after the service executes the provider invokes the consumer to send the defined answer. In the case of the example BP, the answer involves the sending of the actual medical record for the patient, as requested. These two invocations from the consumer to the provider and from the provider to the consumer to answer the invocation, are in general defined asynchronously, although each invocation participating in the interaction can be defined synchronously, for example sending back an acknowledgment of the invocation received to the invoker.

This case of bidirectional interaction is a simple one and its implications are known by the service designers and developer, as they are based on the consumer-provider interaction without specific knowledge of the BP, the domain or the interaction being carried out. However, when the interaction involves more than one request and one answer invocations, the importance of specifying the

choreography for the service becomes more important. SoaML does not define a new element for modeling choreographies as UML sequence diagrams are a good enough fit for the purpose of the choreography specification.

### 7.2.2.3. D3 - Investigate existing services

The main goal of this activity is to reuse the organization's existing services as far as possible. This is done through the use of a central Services Catalogue created in the organization to support the definition and reuse of services. Each time a services development project is being carried out to implement BPs in the organization, after services have been identified, categorized and specified as described in previous activities, the central Services Catalogue has to be searched to find existing services.

The central Services Catalogue keeps a register of all defined services in the organization, describing the functionality each one provides as well as its interface, operations and signature, along with other information which may include, among other data, details about which BPs the service realized, and what the actual location of the service to be invoked is. Before defining the components for implementing the services defined to realize the BP, the Architect searches for services that provide the required functionality or a similar one. Existing services can be used by means of wrappers or adapters to use and/or modify and/or extend existing functionalities, when the functionality provided is not the same as the functionality needed.

The central Services Catalogue grows with each new project, as services that cannot be implemented by reusing existing services are then registered in the catalogue as new ones, to be reused for the implementation of another BP when possible. This registration is also carried out because any future decision on whether to reuse an existing service or build a new one will be taken after searching through the central Services Catalogue.

**Inputs** The Services document with the Identify and categorize and Specify services sections informed, along with the central Services Catalogue with the information about existing services in the organization.

**Outputs** The Services document with the Investigate existing services section informed including the existing services that will be reused, if any, along the central Services Catalogue updated with the information on new services to be developed.

**Roles** The role responsible for this activity is the Architect, but the Analyst also participate in helping to define the reuse of existing services.

**Method** Based on the information about the defined services, described in the Identify and categorize services and Specify services sections from the Services document, the central Services Catalogue is searched by the Architect in order to find existing services as candidates for reuse. If a service exists that fulfills the functionality requirements for the defined service, then the existing service can be used as it is, and the functionality of the adapter will be solely to call the existing component that implements the service; if the functionality provided is similar then the adapter will add or modify the existing functionality to suit the new requirements for the service.

This activity is done manually by the Architect as it is a knowledge-intensive task, but some parts could be automated in the organization by means of the use of semantic systems that are, for example, able to match a description of the functionality of the service to descriptions of existing services, making it easier to discover possible services that may be reused. The services categorization defined in activity D1 - Identify and categorize services will also help in defining a taxonomy for services in the Services Catalogue that may be used, whether manually or automatically. The Architect will also register the new services generated in order to increment the knowledge in the central Services Catalogue.

### Service-oriented modeling with SoaML

The use of SoaML in this activity implies specifying the Participants component (PC) diagram, in which the defined adapter or wrapper for the existing service is shown to relate the defined service to the existing implementation which is being reused. Figure 7.13 shows the reusing of an existing component to provide the appointment for the surgery as defined by the Service “ReceiveRequestForAppointment”. This Service Port defined to provide the service in the Participant “generalHospital” is in turn provided by an adapter that is internal to the Participant, which reuses an existing service to provide the appointment for the patient.

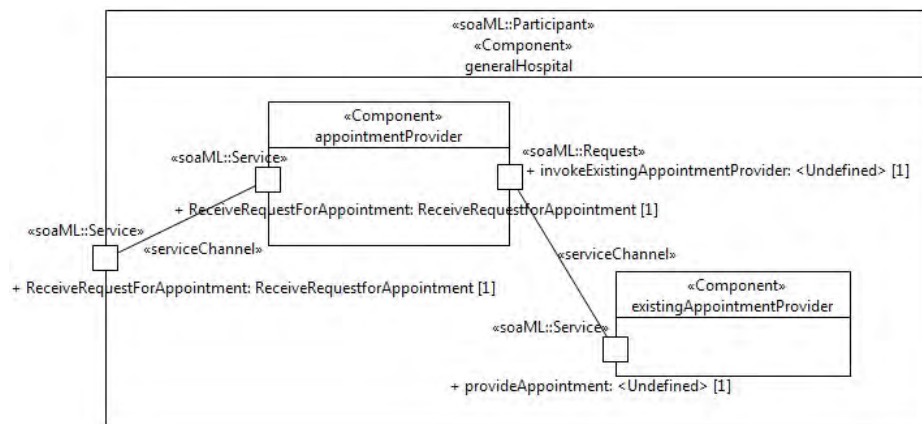


Figure 7.13.: SoaML component reuse for the “ReceiveRequestForAppointment” service

#### 7.2.2.4. D4 - Assign components to services

The components that will implement the defined services must be defined and shown in the components diagram. For each service, it is necessary to define a component with which to implement it. This might be an atomic component, an existing component, or a composite component which uses or has several other internal components. The use of existing components was defined in the previous activity, so those definitions are integrated to provide the complete view for the implementation of services, and the new components that will be implemented are specified.

**Inputs** The Services document with the previous sections informed: Identify and categorize services, Specify services and Investigate existing services. If a Design document is generated by the base software development process for defining design models that is also used here.

**Outputs** The Service document with the section Assign components to services informed. If an Implementation document is generated by the base software development process for defining implementation models this is also updated here to include the implementation information for the defined services.

**Roles** The role responsible for this activity is the Architect, but the Developer also participate in helping define the components to implement the designed services.

**Method** The Architect uses the information concerning existing services found in the central Services Catalogue, as well as the specification of the services defined in the Services document to associate a component to implement the service. The design of components can be performed in several ways, and it is up to the Architect how these will be defined.

### Service-oriented modeling with SoaML

The use of SoaML in this activity implies specifying the Participants component (PC) diagram in which to show the implementation of the defined services, specifying new components to be

generated and, if the component exists, using the definitions for reuse as defined in the previous activity. All the services defined for the Participant are shown in the Participant component and all the internal components needed to implement the services provided are specified. In Figure 7.14 the definition of the component for the service “ReceivePatientMedicalrecord” which is added to the “generalHospital” Participant component presented above, so defining the complete implementation of the Participant.

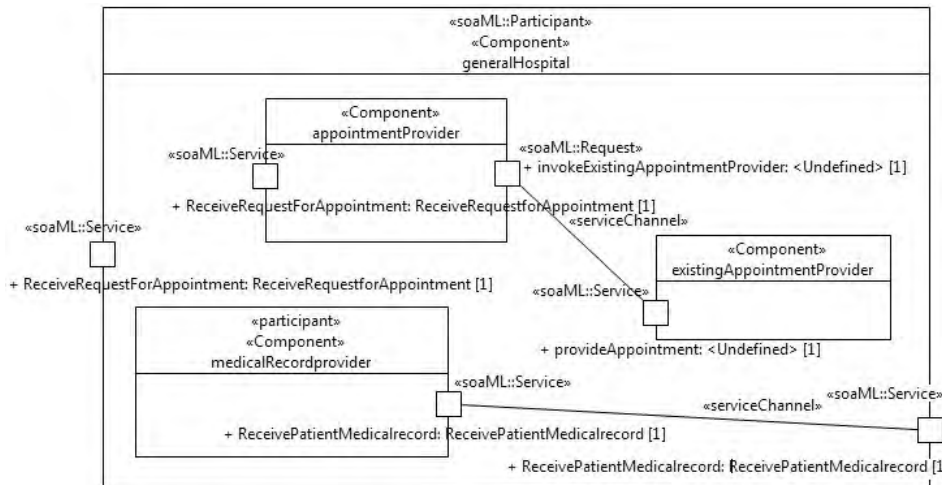


Figure 7.14.: SoaML component definition for the “ReceivePatientMedicalrecord” service

#### 7.2.2.5. D5 - Define services interaction

Services interaction can be defined as the orchestration or choreography of services in the same way as is done for BPs, showing in the first case the interaction of services defined internally for the Participant to provide the defined services, and in the second case showing the interaction with external services from other Participants.

The definition of services interaction helps to get a picture of the interactions by showing in diagram form the interactions that have to occur between the defined services for the BP to be realized. Carrying out this activity is optional as there can be cases in which another view of the interactions defined between participants is not needed, as these are also specified as choreographies attached to each ServiceContract defined.

**Inputs** The Business Process document and the Services document, with all the information for the specification of BP and services informed. Optionally the Requirements document generated by the base software development process can be used to check additional information.

**Outputs** The Services document with the information for the services interaction included. Optionally the Implementation document generated by the software development process can be updated with information about the defined associated components interactions.

**Roles** The role responsible for this activity is the Architect, but the Analyst also participate in helping define the reuse of existing services.

**Method** The interaction between defined services is provided in a UML Sequence diagram showing all the defined services, or various sequence diagrams showing different subsets of services corresponding to different sub-processes in the BP. The messages exchanged, as defined by the ServiceContract for each service, are shown, thus providing the summary of the realization of the BP by services. This overall view can be made by the concatenation of all the choreographies



attached to each defined service, showing them in the order they will be executed when executing the BP. Figure 7.15 shows an example for the services defined between the “patient” and “general-Hospital” participants.

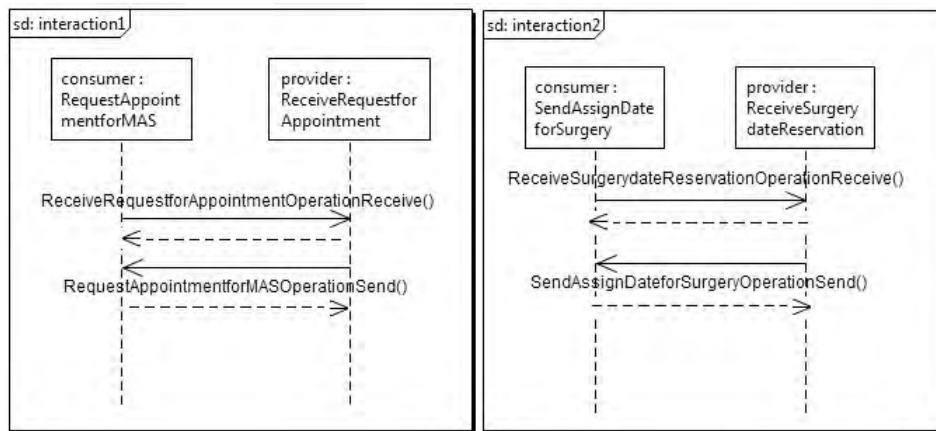


Figure 7.15.: UML sequence diagram showing defined services interaction

### Service-oriented modeling with SoaML

SoaML cannot be used in this activity as it has no corresponding diagram to support it, since the standard does not provide an overall view on the interactions, but only the choreographies attached to each ServiceContract specified for each service defined. Instead, as defined in the activity method the UML Sequence diagram is used to show the interactions between the services defined to realize the BP.

### 7.2.3. Implementation Discipline

The Implementation Discipline aims to develop components to implement the defined services, according to the assignment of services to components performed previously, and defining the issues regarding the implementation and the selected technology. Based on the SoaML service models developed or generated in the Design Discipline, the corresponding code can be constructed manually in the selected technology, or generated automatically by means of MDA engines such as ModelPro<sup>1</sup>.

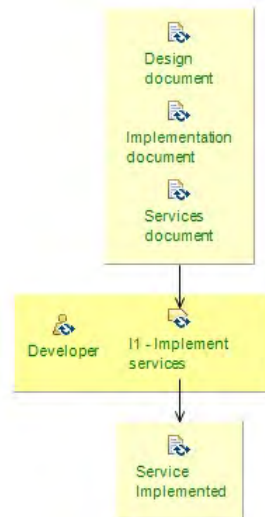
If the code is generated, the logic defining the behavior of the service has to be implemented anyway in the generated code, as only the general structure of the implementation is obtained. For the automatic generation of code we suggest a distribution of Eclipse including the MDA engine ModelPro, and to manually implement them an Eclipse JEE distribution, as described in chapter 9. Figure 7.16 shows the activity detail diagram for the Implementation Discipline.

#### 7.2.3.1. I1 - Implement services

This activity has the objective of implementing the services defined as specified in the SoaML service models obtained in the Design Discipline, and taking into account the type of service, the interfaces designed, the interaction with other services (with or without a repository of services, binding in development or execution time), among others.

For the implementation of services it does not matter whether the SoaML service models have been generated manually or automatically, as long as they are present to guide the implementation. When automatic code generation is performed by means of using the SoaML service models defined, several other steps have to be carried out, which depends on the specific MDA engine used, assigning, for example, other UML profiles for the selected platform such as the “J2EE profile”.

<sup>1</sup><http://www.modeldriven.org/>



**Figure 7.16.:** Implementation Discipline activity detailed diagram

**Inputs** The main input of this activity is the Services document with the correspondence between design and implementation informed. Design and Implementation documents from the base software development process with associated details can be also checked.

**Outputs** The output of this activity is the service implemented in the specified component, to be tested and deployed.

**Roles** The only participating role is that of the Developer who is in charge of the implementation of services.

**Method** To implement the components for the design services the corresponding interfaces and implementation classes have to be coded, providing the operations, parameters and types defined in the design. Services corresponding to interactions with external participants will generally be implemented as Web Services in technologies such as J2EE or .NET, which will be exposed for invocation outside the limits of the organization. For internal services within the organization other options can also be selected such as remote invocation, messages queues, ESB, among others.

When generating the code automatically from the SoaML service models defined, the SoaML service model obtained in the Design Discipline has to be imported into the Eclipse ModelPro distribution and integrated with the modeler and generator of the ModelPro MDA engine. For the generation, an UML deployment diagram is constructed defining the distribution of the components and assigning other UML profiles such as the “J2EE profile” and “Web Service” for the participant component, or if it is implemented manually, several facilities in the Eclipse JEE distribution can be used. These options are discussed in chapter 9.

### 7.3. BPSOM phases

The BPSOM reference lifecycle proposed is iterative and incremental and follows the definition of the Unified Process [Jacobson et al., 1999] although as mentioned in section 7.1 it can be added to any base software development process that the organization uses. This BPSOM lifecycle is only proposed as a guide to indicate the emphasis on the carrying out of activities at each stage of development. Depending on the base software development process used, the phases can be different, but using this guide the conceptual definition for performing the different activities can be analyzed and integrated accordingly.

BPSOM phases are: Inception, Elaboration, Construction and Transition, and the carrying out of activities defined in each one has to be inserted and integrated in the phases of the base software development process in the organization, making the corresponding adaptations when needed. The BPSOM lifecycle is defined to navigate from the Design&Anaylisis phase to the Configuration phase guiding the implementation of BPs by means of services. In Figure 7.17 the relationship between BPCIP and BPSOM phases is shown.

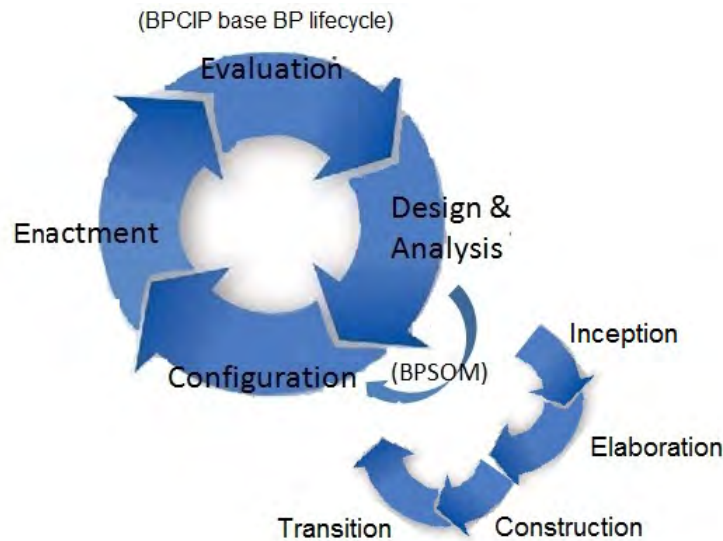


Figure 7.17.: Relationship between BPCIP and BPSOM phases

### 7.3.1. Inception phase

The BPSOM Inception phase puts the emphasis of development on the Business Modeling Discipline, and the first activities defined in the Design Discipline can also be performed, as BP models are available. The meetings with the business area people are fundamental for gathering the information that is needed about the organization and its BPs, as well as the requirements for the system development. The activities in the Requirements Discipline from the base software development process are also carried out to specify the requirements.

The Business Modeling activities carried out in this BPSOM phase also correspond to the BPCIP Design&Analysis phase in which BPs have to be modeled and validated and verified when needed. The Requirements and Design activities, however, as software Disciplines, correspond to the BPCIP Configuration phase in which the service oriented system is being developed. So, in BPSOM Inception phase several iterations can be performed between the Analysis&Design and Configuration phases of BPCIP, to model BPs, specify requirements for the system and carry out an initial identification of services. In Figure 7.18 the activity diagram for the BPSOM Inception phase is presented.

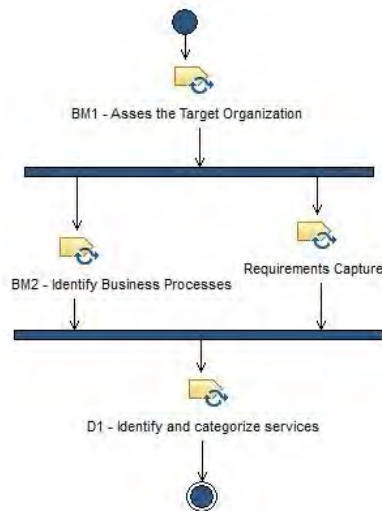


Figure 7.18.: BPSOM Inception phase activity diagram

### 7.3.2. Elaboration phase

The BPSOM Elaboration phase is focused on the design of the service oriented system, by identifying and specifying the services needed, although the modeling of BPs can also continue to refine the BP models or to add new information or new models, along with the specification of requirements for the system. The implementation of services can begin once the first services are identified and specified, based on the available SoAML service models. The implementation of services is also tested following the verification activities defined in the base software development process.

The Design and Implementation activities carried out in this phase correspond to the BPCIP Configuration phase, where the focus is on implementing the BPs by means of services. Although some activities from the BPCIP Design&Analysis phase can still be performed, such as modeling BPs and specifying requirements for the system, most of the BPs and requirements have been already modeled and specified in the BPSOM Initial phase.

So in BPSOM Elaboration phase some initial iterations can be performed between the Analysis&Design and Configuration phases of BPCIP, and the rest of the iterations will only carry out design and implementation activities corresponding to the BPCIP Configuration phase. Figure 7.19 shows the activity diagram for the BPSOM Elaboration phase.

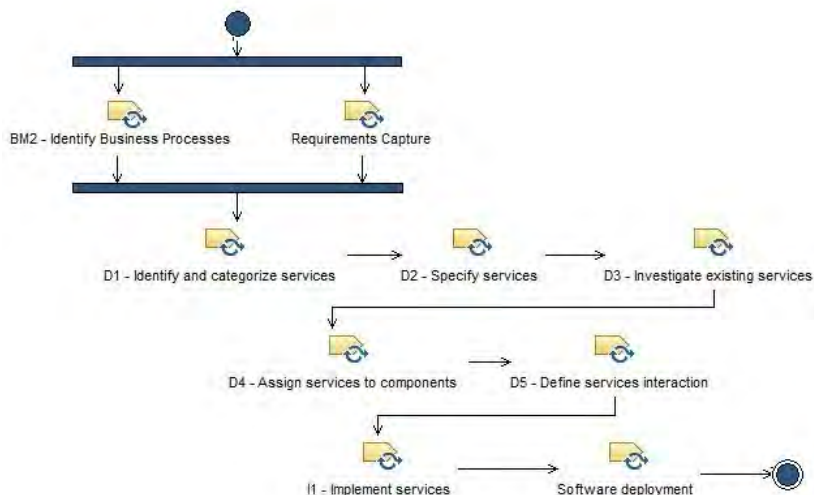


Figure 7.19.: BPSOM Elaboration phase activity diagram

### 7.3.3. Construction phase

In the BPSOM Construction phase the focus is on the development of the services defined to support the modeled BPs, although many services continue to be specified in this phase too, as some are being implemented and others are being first designed and then implemented, in an iterative and incremental way.

Services implementation is also tested following the verification activities defined in the base software development process. The Design and Implementation activities carried out in this BPSOM phase are part of the BPCIP Configuration phase where the service system to realize the BP is being developed. Figure 7.20 shows the activity diagram for the BPSOM Construction phase.

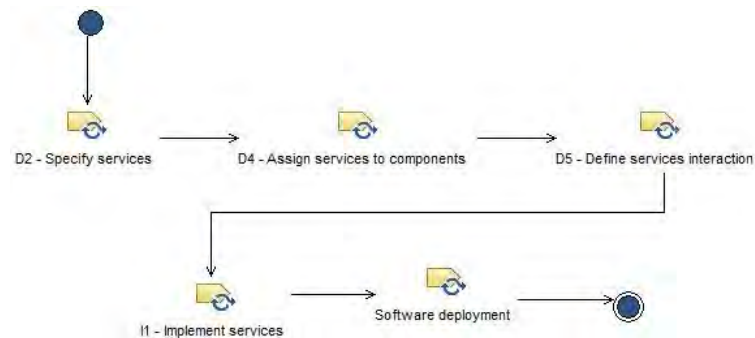


Figure 7.20.: BPSOM Construction phase activity diagram

### 7.3.4. Transition phase

The BPSOM Transition phase aims to deploy the service oriented system developed in the chosen infrastructure, where the implemented services will be invoked from the BPs executed in a process engine. The activities defined in the base software development process for carrying out the deployment are performed, including for example, the testing of the services deployed in the testing environment to ensure their correct function when interacting with the process engine.

The executable BP has been obtained as output of the corresponding BPCIP Configuration activity defined to do this. As there are no specific activities defined in BPSOM for testing or deploying services because they are part of the base software development process, no diagram is presented for this phase. After completing this phase of BPSOM the BPCIP Configuration phase is also completed, and the BP is then executed in the organizational environment performing the activities corresponding to the BPCIP Enactment phase.

## 7.4. EPF implementation

As with BPCIP, the BPSOM methodology is also implemented as an EPF Composer method plug-in to provide interoperability with other processes defined in the same way, so it can be integrated easily in any base software development process. It is also published as a web site by means of the EPF Composer facilities to provide easy access to it, so it can also be used although the base software development process is not specified as method plug-in. In Appendix B the complete BPSOM Web site is presented.

### 7.4.1. BPSOM method plug-in

The BPSOM method plug-in defines the previously presented BPSOM elements in the EPF Composer, and once all the elements are in place the export option enables BPSOM to be exported as a method plug-in, and the publish option allows BPSOM to be published as a web site. In Figure 7.21 a screenshot of the definition of BPSOM in the EPF Composer is shown.

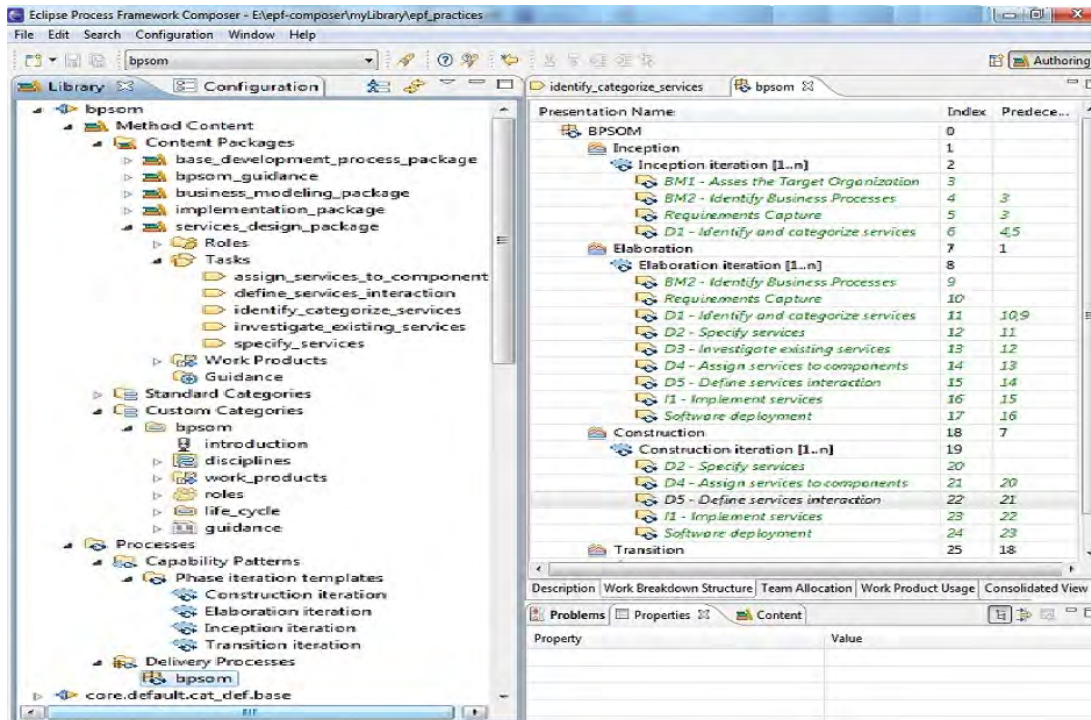


Figure 7.21.: Example of BPSOM method plug-in definition in EPF Composer

On the left side the tree of elements defined can be seen: method content with categories and elements, and processes with templates and delivery process. On the right side the definition of the delivery process with the phases defined and the activities included in each phase, the defined sequence of activities, among other information. The BPSOM method plug-in can be downloaded from the MINERVA BPSOM Web site Delgado [2010-2011] and then imported into an EPF Composer to be added or integrated in the base software development process or to be used in any way that is wished.

#### 7.4.2. BPSOM web site

The BPSOM methodology is easily accessed through the MINERVA Web site where it is published Delgado [2010-2011], providing easy access and use of the defined elements. In Figure 5.13 a screenshot of the BPSOM web publication is shown, on the left side the categories defined can be seen: Disciplines, Work products, Roles and Lifecycle, along with some of its elements that it is made up of, such as activities, tasks, deliverables, roles. On the right side an example of tasks definition is presented for BM2 - Identify Business Processes, showing participating roles, work products defined as inputs and outputs, purpose, description and the Discipline to which it belongs.

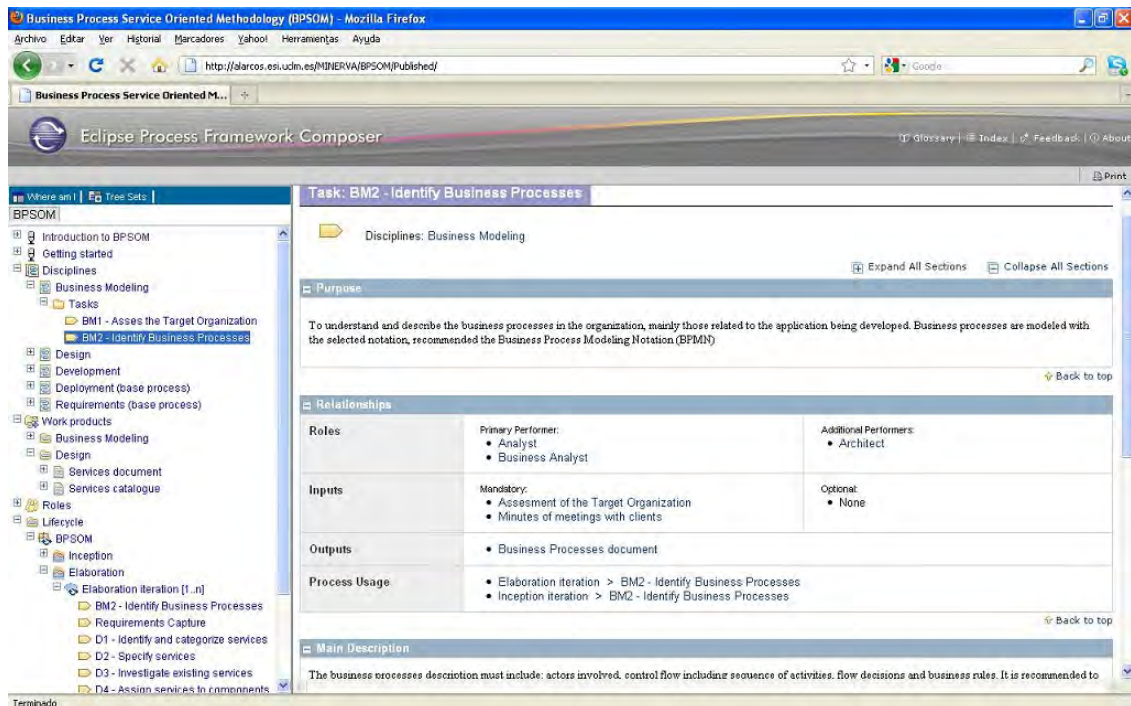


Figure 7.22.: Global view of BPSOM web site created using EPF composer

The EPF Composer generated BPSOM Web site provides easy navigation from one element to all the other elements related to it, by means of generated links between them. In addition the diagrams associated with activities, roles, work products and iterations defined are generated and available as part of the definition of the corresponding elements. The Web site provides an easy guided way to use BPSOM methodology in developing service systems from business processes.

## 7.5. Conclusions

This Chapter has presented the BPSOM methodology which provides a guide for developing service-oriented systems from BPs in the organization. The three Disciplines it defines have been presented along with the defined activities, input and output artifacts, participating roles, a description of the method to perform the activities, and the use of BPMN2 and SoAML standards to model BPs and services, respectively. It also provides automatic generation of SoAML service models from BPMN2 models which is described in chapter 8.

For each Discipline a detailed activity diagram is presented summarizing the information about the activities and other elements it is made up of. The BPSOM phases proposed to guide the use of the methodology were also presented, showing their relationship with the BPCIP phases that includes them, as BPSOM is only defined for implementing BPs with services, and BPCIP is defined for the management and improvement of BPs through their complete lifecycle. An activity diagram for each BPSOM phase was also provided giving a general view of the flow between activities in each phase.

Finally the implementation of BPSOM as an EPF Composer method plug-in was presented, which allows the integration and use of BPSOM with the base software development process used in the organization in an easy and effortless way. The corresponding Web Site generated from it was also described, which provides easy access for its use throughout the organization.





## Chapter 8.

# Generation of SoaML models from BPMN2 models

This Chapter presents the model-driven approach defined in MINERVA framework and integrated into the BPSOM methodology, for the automatic generation of SoaML service models from BPMN2 models.

The Chapter is organized as follows: in section 8.1 the model-driven approach is presented, in section 8.2 the integration of the specific steps defined for service models generation into BPSOM is described and in section 8.3 the correspondences defined between the BPMN2 and SoaML meta-models are presented. In section 8.4 the transformations defined are presented and in section 8.5 an example of the application of the model-driven approach is provided, finally in section 8.6 conclusion for the chapter are discussed.

The contents of this chapter complement the contents in chapter 7 which describes the BPSOM methodology for service oriented development from BPs in which the model-driven approach is integrated.

### 8.1. Introduction

The model-driven approach defined in MINERVA for the generation of SoaML [OMG, 2009b] service-oriented models from BPMN2 [OMG, 2011a] BP models follows the MDA [OMG, 2003] principles mentioned in chapter 3, and it is based completely on the use of the OMG standards: BPMN2, SoaML and QVT [OMG, 2008c]. As presented in chapter 4 we have integrated the metamodels corresponding to the defined standards as well as the needed tool support, based on the definition of the Eclipse MINERVA design distribution which integrates existing plug-ins and the new ones developed. In Figure 8.1 the MDA vision of MINERVA framework is presented.

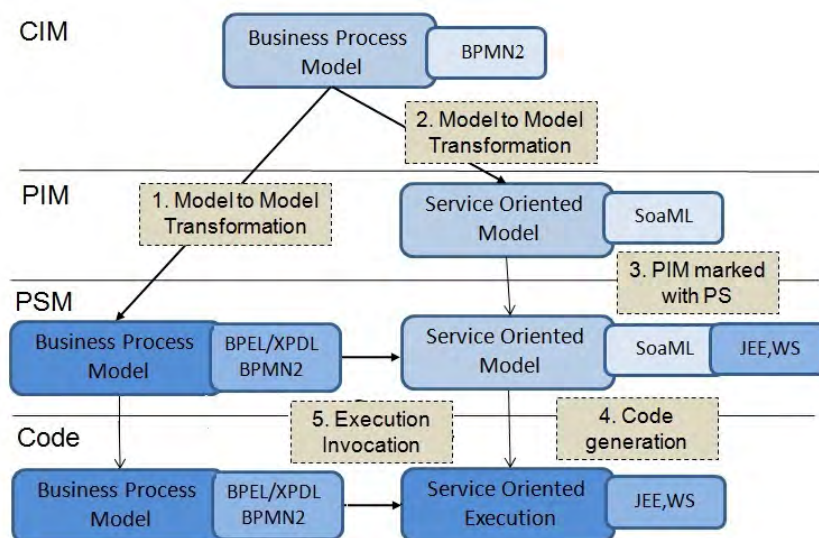


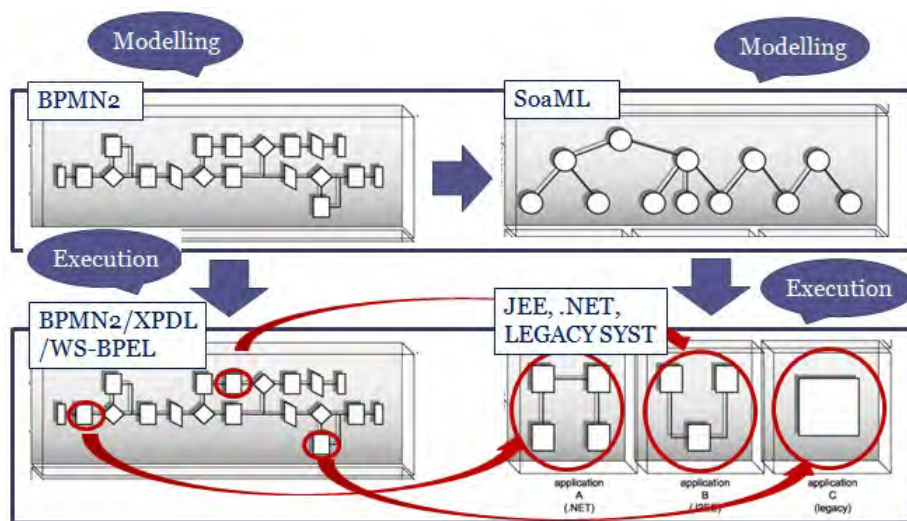
Figure 8.1.: MDA vision in MINERVA framework

The BPMN2 model constitutes the Computation Independent Model (CIM) which is the initial input for all the transformations and steps defined. Going down the right side of Figure 8.1 the SoaML service model constitutes the Platform Independent Model (PIM). From the PIM SoaML model the Platform Specific Model (PSM) is obtained by adding platform specific information, such as that provided by the JEE and WS profiles of the MDA engine ModelPro, to generate the associated code. From the SoaML service model the code can also be generated manually, as it provides the design information needed for implementation.

On the left side of Figure 8.1, the BP will be executed in a process engine in a suitable language (XPDL [WfMC, 2008]/ BPEL [OASIS, 2007]/ BPMN2) being both, the PSM and the associated code, obtained from the same BPMN2 model, which also constitutes the PIM by means of the Identity transformation (not included in Figure 8.1). To obtain the XPDL/BPEL/BPMN2 to be executed two cases have to be taken into account: firstly if the execution language is XPDL/BPEL the BPMN2 has to be transformed into the selected language by means of using existing tools, secondly if the execution language is also BPMN2 no transformation is needed.

To add invocations to the generated services into the executable BP model, taking the XPDL/BPEL/ BPMN2 model as input, we insert the corresponding invocation into the service activities definition, using the information provided in the SoaML model. Note that several other elements have to be added and/or defined in the BPMN2 to make it executable, such as, among others, implementing user forms, scripts, for which manual work is required.

In this way, on one hand the services are generated, and on the other hand, the executable BP model is generated, which contains the invocation to the generated services. The chain of transformations presented makes it possible to complete the traceability from the BP to its services implementation, providing the information on which activity from the BP model is designed as a service in the SoaML model, as well as the correspondence in the executable BP model between activities with the implemented services. This is illustrated in Figure 8.2, excluding intermediate steps.



**Figure 8.2.:** Business processes and services transformations vision of MINERVA

The model-driven approach integrated into BPSOM is used in the BPCIP lifecycle to navigate from the Design&Analysis phase to the Configuration phase as presented in chapter 7, as part of the guide to develop service-oriented systems from BPs, and by executing the defined transformations to generate service models automatically. In Figure 8.3 the method of work of MINERVA as presented in chapter 4 is shown, to place the reader in the context of the definitions that are described below. The use of the model-driven approach corresponds to the steps 2 and 3 marked in red circles in BPSOM, in the rectangle between BPMN2 models and SoaML service models.

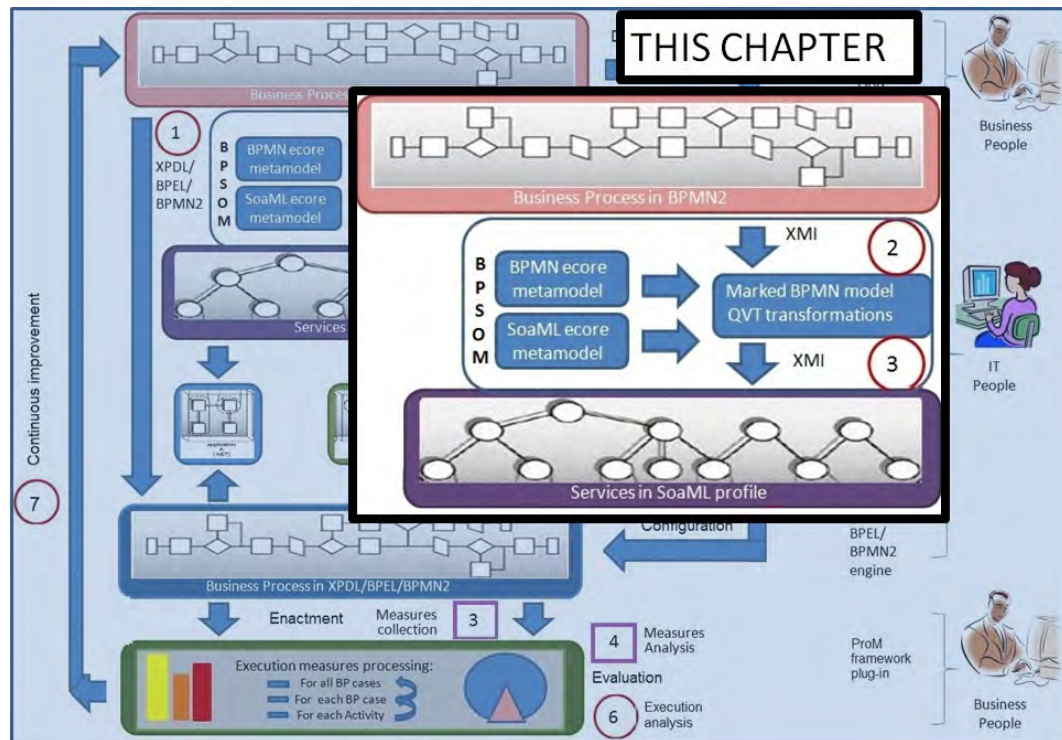


Figure 8.3.: MINERVA method of work through BPCIP

## 8.2. Integration in BPSOM

In chapter 7 the integration of the model-driven approach for the automatic generation of services was indicated in each corresponding activity of the Design Discipline. In this section we present the specific steps that have to be carried out for the automatic generation, as shown in Figure 8.4, relating them to the Disciplines and corresponding activities. As presented in chapter 7 the defined activities can also be carried out on a manual basis, that is, without applying the transformations, as the rules defined also form a guide for the design of the services to realize the BP.

As is shown in Figure 8.4 the “Identify Business Processes” activity from the Business Modeling Discipline includes the specification of the BPMN2 model by the Business and Software area in the selected modeler, saving it in BPMN2 format. The Design activities “Identify and categorize services” and “Specify services” in which SoaML service models are generated, need as a previous step to add the identification of ServiceTasks by the Architect (and other elements for the generation if the labels to be generated want to be controlled), and to import the BPMN2 model which is in BPMN2 format into the Eclipse MINERVA design.

The generation of the SoaML service models is done by executing the defined QVT transformations in Eclipse MINERVA design, with the BPMN2 model as input and by generating the SoaML model as output. Finally, the implementation of services includes the generation of code from the SoaML service model (which can also be obtained manually), which is then completed by the developers adding the corresponding logic into the services, and also adding the needed code in the BP model to make it executable and to invoke the services generated.

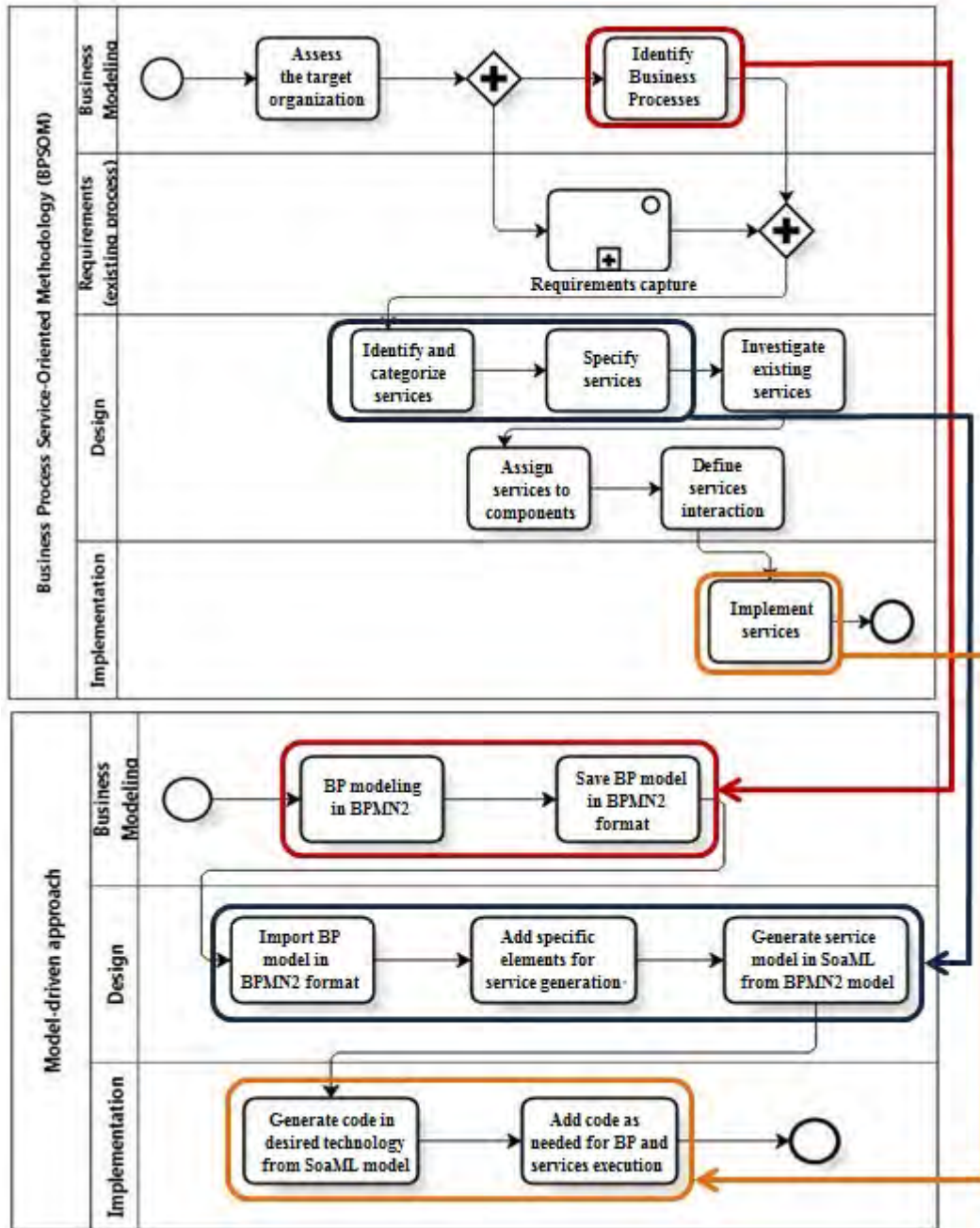


Figure 8.4.: Model-driven approach integrated in the BPSOM methodology

The procedure described above, presented the high level steps that have to be performed and their relationship to BPSOM Disciplines and activities, where technical details were hidden. In Figure 8.5 the procedure is presented including the technical details, organized in four groups of related steps with a focus on the chain of transformations from the BPMN2 model to the SoaML service model: (1) to obtain the BPMN2 model, (2) to obtain its XMI representation, (3) to generate the SoaML XMI file from it and (4) to import the SoaML model for visualization. Here we only briefly mention the tools supporting the generation procedure, these are described in chapter 9.

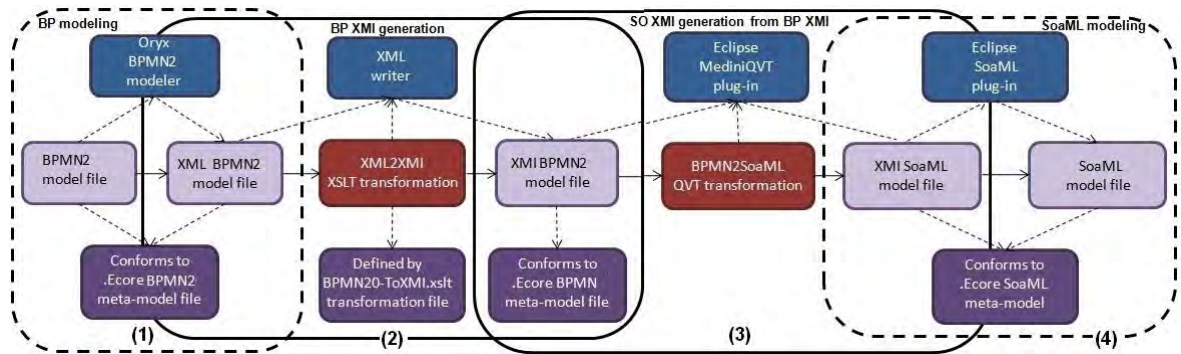


Figure 8.5.: Step by step generation procedure and tool support

The first two steps consist in the modeling of the BP in BPMN2, which in turn represents the input of the whole automated procedure. This BPMN2 model conforms to the BPMN2 metamodel provided by OMG, and is saved in the XML BPMN2 format. We use the Oryx editor <sup>1</sup> for the BP modeling as it enables the model to be saved in the required format, but any other modeler that provides this functionality can be used (such as Activiti modeler <sup>2</sup> which is based on the same editor as Oryx).

The Architect will then add design information to the BP model to generate the service model, as explained later. The Eclipse MINERVA design provides all the plug-ins and tools needed to support this generation procedure. The Eclipse BPMN2 SAP editor <sup>3</sup> allows the BPMN2 model to be visualized and validated against the metamodel, prior to executing the generation. The QVT transformations have as their input the XMI file representing the BPMN2 model, generating as output an XMI file representing the SoaML model.

That being so we need to transform the input model into the XMI format. Using the XSLT transformation provided by OMG in [OMG, 2011a], the XMI file of the BPMN2 BP model is obtained, which is also compliant to the BPMN2 metamodel. Once the XMI file of the BP model is in place the QVT transformations to generate the service model are executed. The Eclipse MediniQVT <sup>4</sup> plug-in is integrated to define and execute the QVT transformations. It requires the metamodels for BPMN2, UML and SoaML in .ecore format from EMF <sup>5</sup> to be registered in the environment prior to the generation.

Three main transformations for the generation of services from ServiceTasks are provided based on the design options defined by the SoaML standard regarding the communication patterns presented: bidirectional and unidirectional with UML simple Interfaces, and bidirectional with ServiceInterface. The Architect therefore only has to select the transformation he/she wants to execute, from the ones provided in Eclipse MINERVA design. A summary of the high level steps and the defined transformations is shown in Figure 8.6.

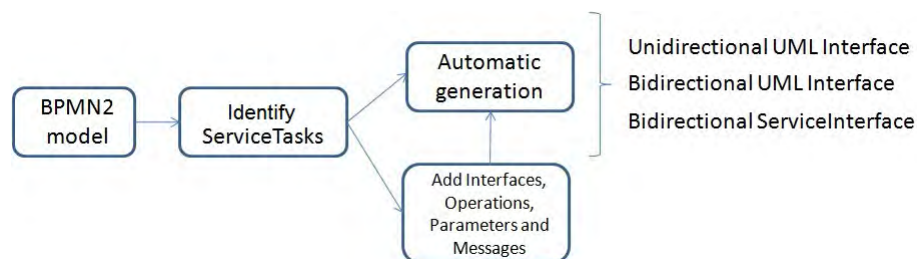


Figure 8.6.: Service design generation options provided in BPSOM

In the automatic generation option the generation is based only on the identification of ServiceTasks in the BPMN2 model. The generated Services, Participants, Interfaces, Operations, Parameters

<sup>1</sup><http://bpt.hpi.uni-potsdam.de/Oryx/Research>

<sup>2</sup><http://www.activiti.org/components.html>

<sup>3</sup><http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/c04f0691-0a76-2d10-1098-ec518f7bdf68>

<sup>4</sup><http://projects.ikv.de/qvt/>

<sup>5</sup><http://eclipse.org/modeling/emf/>

and MessageTypes are all given the names of the associated activities in the BPMN2 model, corresponding to the provider and consumer. Three other transformations were defined in case the Architect wants to have control over the names for the generated artifacts by adding Interfaces, Operations, Parameters, and Messages into the BPMN2 model, which are provided optionally as the effort of defining these elements in the BPMN2 model is considerable.

After the transformations are executed the SoaML XMI file which conforms to the SoaML meta-model provided by OMG is generated. The SoaML XMI file is then imported into the Eclipse SoaML <sup>6</sup> plug-in we have developed, to be able to visualize the generated service model.

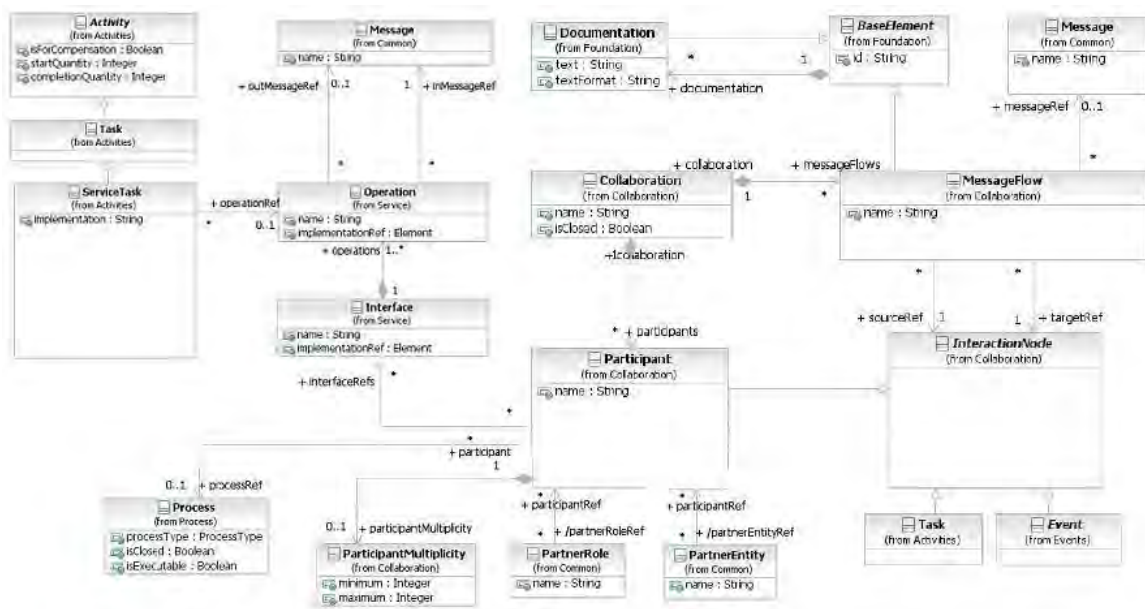
### 8.3. BPMN2 vs SoaML correspondences

Based on the sub-ontologies presented in chapter 4 the elements in both BPMN2 and SoaML metamodels that we want to relate to generate service models from BP models were identified. In this section the correspondences defined are presented.

#### 8.3.1. BPMN2 key elements in transformations

As presented in chapter 3 BPMN2 organizes BP modeling around four main constructions: Process, Collaboration and Choreography, and Conversation, which is a particular use of a Collaboration, as well as an informal description of this. All these elements are connected, when present, to the Definitions element, which is the outermost containing object for all BPMN elements.

We concentrate on the collaboration construction of BPMN2, as it represents the inter-organizational definition of the interaction needed between several organizations following a common business goal. Services are modeled explicitly too by integrating several elements such as Interfaces, Operations, Messages in and out and their relationships, although these facilities are not used for the automatic generation. In Figure 8.7 some of the core elements of the BPMN2 metamodel for Collaboration, Messagesflow, Participants, Activities and ServiceTask modeling we use in our approach are shown.



**Figure 8.7.:** BPMN2 [OMG, 2011a] metamodel elements used in the correspondences to SoaML

In Figure 8.7 it can be seen that a ServiceTask activity, which we will transform into a provided service, can be related to Operations with Messages as in and out parameters, which are integrated

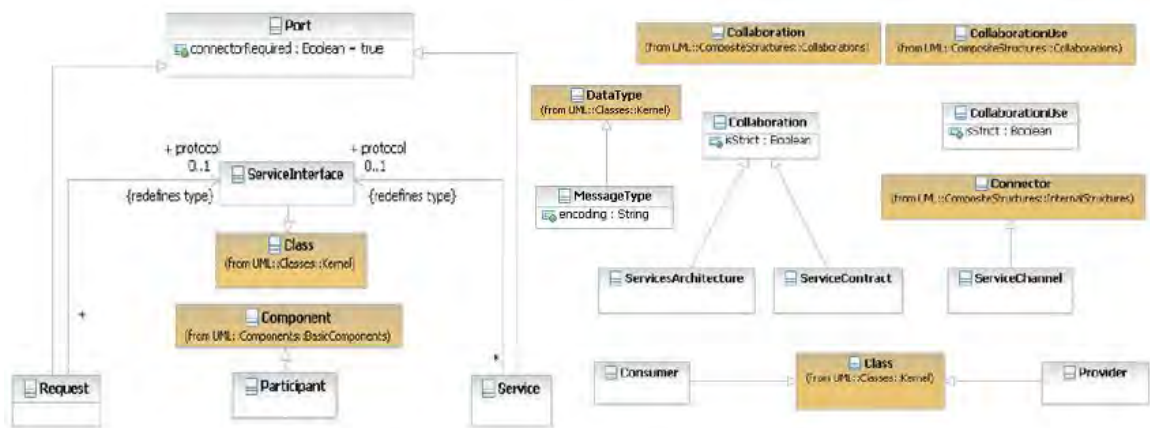
<sup>6</sup><http://alarcos.esi.uclm.es/MINERVA/TOOLS/soamlPlugin.htm>

in an Interface referenced by a Participant. If these elements were present in the BPMN2 model the transformation to be executed has to be selected from the ones that takes them into account. A Collaboration is made up of Participants, each of which refers to its associated Process, along with the Messagesflows between participants, which have a sourceRef and a targetRef referring to interactionNode elements, which can be a Task, an Event or a Participant.

### 8.3.2. SoaML key elements in transformations

As presented in chapter 3 key elements for modeling services in this approach are based on the concept of specifying Participants which provides Services to be used from other participants, and which consume services from other participants. Service and Request Ports are defined to do that, which are then typed by corresponding Interfaces.

A ServiceContract specifies the roles defined to interact with the Service, together with a Choreography illustrating its behavior, if needed, the Service is specified by Interfaces, Operations, Parameters and their types, pre and post conditions for the use of the operations, if required. For all the Participants involved, the ServicesArchitecture shows the ServiceContracts in which they play a role, along with the associated roles and Ports provided and consumed. In Figure 8.8 key elements of the SoaML metamodel for service modeling are shown.



**Figure 8.8.:** SoaML [OMG, 2009b] metamodel elements used in the correspondences to BPMN2

As can be seen in Figure 8.8, both **ServiceArchitecture** and **ServiceContract** are UML **Collaborations**, which define the elements they have to provide: **Participants** parts, roles, and **CollaborationUse**, to show the participation in existing collaborations. **Consumers**, **Providers** and **ServiceInterfaces** can be a UML class or Interface, and participants can be UML classes or components, although not all these derivations are shown in the metamodel, but rather in the stereotype definition.

### 8.3.3. BPMN2 vs SoaML correspondences definition

To specify the QVT transformations the correspondences between elements from the BPMN2 and SoaML metamodels were defined. These correspondences specifies which element/s in the source model -which conforms to the source metamodel- will be transformed into which element/s in the target model -which conforms to the target metamodel- when the transformation is executed. This corresponds to the model transformation definition we use in MINERVA as presented in chapter 3. Based on the rationale for the ontology definition, these correspondences between elements in both metamodels were defined prior to the definition of the QVT transformations, which are shown in Figure 8.10.
















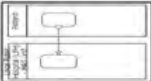






BPMN 2.0		SoaML beta2	
Element	Icon	Element	Icon
Definitions	(complete model)	Model	 <Model>
Process		Participant	 <Class>
ServiceTask (provider) +MessageFlow (sourceRef = Task , targetRef = ServiceTask) +Task (consumer)		MessageType	 <Class>
		Interface	 <Interface>
		Operation	 <Operation>
		Parameter In/Out	 <Parameter>
		ServiceContract Provider Consumer	 <Collaboration>  <Property> provider  <Property> consumer
ServiceTask (provider)		Service (of Participant)	 <Port>
Task (consumer)		Request (of Participant)	 <Port>
Collaboration		Services Architecture	 <Collaboration>
Collaboration Participant		Participant Part	 <Property>
MessageFlow (sourceRef = Task , targetRef = ServiceTask)		CollaborationUse Dependency consumer provider	 <Collaboration Use>  <Dependency>

Figure 8.9.: Key correspondences between BPMN2 and SoaML metamodels for ServiceTask

In the first place, the service *Model* element corresponds to the *Definitions* element from BPMN2 , which encloses the rest of the elements. The *Process* element corresponds to a *Participant* which is a UML Class. A *MessageFlow* between two different *Processes* where the *targetRef* is a *ServiceTask* defines the specification of the service. From this construct we generate the *Interface* for the *Provider* (and another for the *Consumer* in the bidirectional case) with *Operations*, *Parameters* in and out typed with the generated *MessageType* as UML class.

We set the *Parameters* type of the *Operations* as being of the *MessageTypes* we have generated, following a good design practice, which allows encapsulation of the *Parameters* in the *MessagesType*, instead of listing them as part of the *Operation* signature. This in turn allows us to reduce the impact of parameter changes, both in quantity or type. We also generate the associated *ServiceContract*, which is a UML *Collaboration* in which the *Provider* and *Consumer* roles are specified, the *Provider* being the *targetRef* pointing to the *ServiceTask* and the *Consumer* being the *sourceRef* pointing to the associated *Task*.

The identification of a *ServiceTask* also defines a *Service Port* which is a UML *Port* that is typed by the *Interface* the service provides (*Provider* role), and is associated to the *Participant* generated from the *Process* that owns the *ServiceTask*. The *Task* which is associated to the *ServiceTask* by a *MessageFlow* maps to the *Request Port*, which is typed by the consumed *Interface* (*Consumer* role), and it is associated to the *Participant* generated from the *Process* owning the *Task*.

The BPMN2 *Collaboration* (collaborative process) itself is mapped to the *ServicesArchitecture* which is a UML *Collaboration*. The elements of the UML *Collaboration* are defined by associating the *Participants* into *Participants* parts properties of the *Collaboration*, and the UML *CollaborationUse* to the UML *Collaboration* defining the *ServiceContract* of the associated *Service*. The UML *Dependencies* reference to the generated roles *Provider* and *Consumer* of the *Service*.



When in addition to the *MessageFlow* and *ServiceTask*, the *Interface*, *Operations*, *Parameters* and *Messages* for the *Service* provider are present in the BPMN2 model, we used these elements to generate the corresponding ones for the service. The correspondences that change for this case are shown in Figure 8.10, the ones that are not shown remain as defined previously.

BPMN 2.0		SoaML beta2	
Element	Icon	Element	Icon
MessageFlow (sourceRef = Task , targetRef = ServiceTask)		ServiceContract Provider Consumer	<Collaboration> <Property> provider <Property> consumer
Message		MessageType	<Class>
Interface	Interface	Interface	<Interface>
Operation	Operation	Operation	<Operation>
MessageRef In/Out		Parameter In/Out	<Parameter>

**Figure 8.10.:** Key correspondences between BPMN2 and SoaML metamodels for ServiceTask and other elements

## 8.4. QVT transformations

QVT transformations are defined with the QVT Relations language, as presented in chapter 3, by defining template objects in the BPMN2 source domain and in the SoaML target domain. As the SoaML metamodel references elements from the UML metamodel which is included, UML elements are also accessible from the rules. The rules defined in the QVT transformations allow the elements in the target SoaML model to be generated as defined in the correspondences, as well as the relationships between the generated elements in the resulting model. In this section the QVT transformations and rules defined are presented.

### 8.4.1. General definitions

To generate the SoaML elements defined, the construction of UML elements is enforced first and then the corresponding SoaML stereotypes are applied to each of them. The SoaML profile being applied is specified in the main rule as an attribute of the UML model. We have also defined a package structure to organize the resulting SoaML service model, grouping the elements generated in four packages: Participants, Messages, Services and ServicesArchitecture.

In the first one, all the Participants generated are grouped, in the second one all the generated Messages, in the third one we create an inner package for each Service, and within this all the elements for the Service are grouped: Interfaces, ServiceContract and Usages. In the fourth package all ServiceArchitecture elements are grouped, including CollaborationUses and Participants parts referencing the elements created. The general algorithm defines the logical creation of packages and the application of SoaML stereotypes, as shown in the pseudo code in Figure 8.11.

```

1 Procedure bpmn2soaml(bpmodel:bpmn2):(somodel:soaml)
2   Create uml:Model from bpmn2:Definitions, apply stereotype soaml:SoaMLPackage;
3   Create uml:Package Participants, apply stereotype soaml:SoaMLPackage;
4     for each bpmn2:Process in Definitions do
5       create uml:Class; apply stereotype soaml:SoaMLParticipant;
6     end for {Participants};
7   Create uml:Package Messages, apply stereotype soaml:SoaMLPackage;
8     for each bpmn2:MessageFlow with targetRef=ServiceTask and sourceRef=Task (or Message is present) do
9       create uml:Class; apply stereotype soaml:Message;
10    end {Messages};
11  Create uml:Package Services, apply stereotype soaml:SoaMLPackage;
12  for each bpmn2:MessageFlow with targetRef=ServiceTask ServiceTask (with or without interfaces operations, parameters)
13    and sourceRef=Task do
14    Create uml:Package ServiceTask.name+Service, apply stereotype soaml:SoaMLPackage;
15    create uml:Interface for provider apply stereotype soaml:Provider (always;
16    create uml:Interface for consumer apply stereotype soaml:Consumer (when bidirectional);
17    create uml:Class for service interface apply stereotype soaml:ServiceInterface (when bidirectional
18    create uml:usages between interfaces (and realization when ServiceInterface);
19    create uml:Collaboration for contract, apply stereotype soaml:ServiceContract;
20    define uml:rol provider (interface provider), apply stereotype soaml:Provider;
21    define uml:rol consumer (interface consumer), apply stereotype soaml:Consumer;
22    create connector between provider and Consumer; apply stereotype soaml:ServiceChannel;
23  end for {Services};
24  for each participant, message and service do
25    type provider and consumer parameters in associated interfaces with generated messages;
26    add Ports to participant for services provided and consumed; apply stereotypes soaml:Service and soaml:Request;
27  end for {update definitions};
28  create uml:Collaboration from bpmn2:Collaboration apply stereotype soaml:ServiceArchitecture
29    for each provider and consumer created do
30      create uml:ParticipantPart typed with the corresponding Participant
31    end for {ParticipantPart};
32    for each service created do
33      create uml:CollaborationUse typed with corresponding ServiceContract
34      create uml:rolebindings between provider and consumer roles in the ServiceContract;
35    end for {collaborationUse};
36 end {Procedure};

```

Figure 8.11.: General Algorithm for services generation pseudo code

In the first place, we generate the Model from Definitions, and then we generate Participants from Process, and MessagesTypes from Messages or MessageFlows and elements involved (when Messages are not present), as these are the base elements of the generation process. To generate services we provide different options depending on the transformation being applied, as mentioned before: marking only the activities for service generation as ServiceTasks or adding elements for specifying the service provider such as Messages, Interface, Operation and Parameters following the guides we provide.

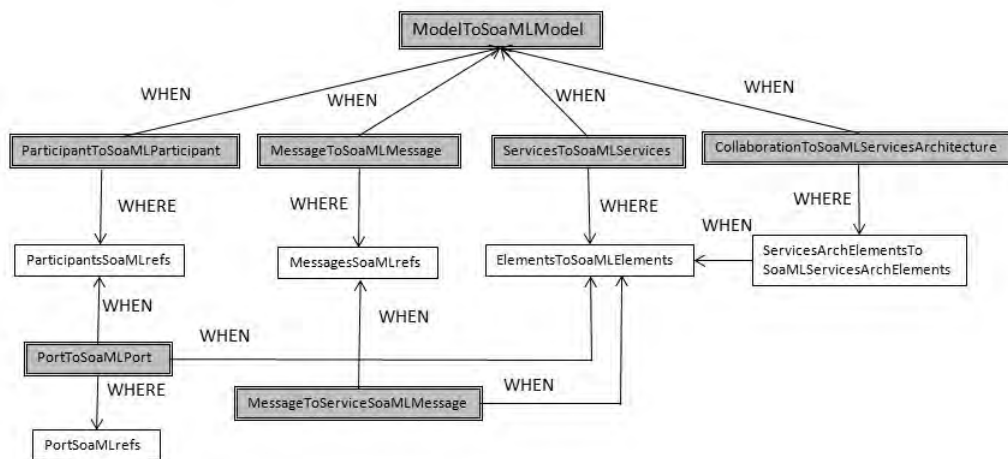
After the Participants, Messages and Services are generated, we assign the Messages as the types of the parameters in the generated operations, and create Ports on the Participants generated, typed with the corresponding Service or Request stereotype, depending on the service being provided or consumed. When all the generation has been completed for the base elements, the ServicesArchitecture is created, as it references the elements generated. The corresponding UML Collaboration is created along with the CollaborationUses referencing the ServiceContract of each Service, the Participants involved and the roles they play.

The general algorithm we have presented in Figure 8.11 shows the logical defined construction for generating services, which is formalized in the QVT relations shown in Figure 8.12. These QVT relations define a hierarchy for the execution of the transformation, as well as several dependencies between the relations that are stated in the WHEN and WHERE sections of each rule. This hierarchy is also shown in Figure 8.12 where the top relations are shown in grey and the invoked relations are shown in white.

```

transformation bpmn2uml2soam12 (bpmn:bpmn2, soam1:soam12){
  top relation ModelToSoaMLModel {...}
  top relation ParticipantToSoaMLParticipant {...}
  relation ParticipantsSoaMLRefs {...}
  top relation MessageToSoaMLMessage {...}
  relation MessagesSoaMLRefs {...}
  top relation ServicesToSoaMLServices {...}
  relation ElementsToSoaMLElements {...}
  top relation MessageToServiceSoaMLMessage {...}
  top relation PortToSoaMLPort {...}
  relation PortSoaMLRefs {...}
  top relation CollaborationToSoaMLServicesArchitecture {...}
  relation ServicesArchElementsToSoaMLServicesArchElements {...}
}

```



**Figure 8.12.:** Hierarchy and dependencies between the QVT relations defined

In each QVT relation, the checkonly domain verifies the existence of the elements in the BPMN2 model, then two enforce domains are defined: the first enforce domain creates the corresponding base UML elements and the second enforce domain applies the SoaML stereotypes to the base UML elements created. This pattern of checkonly BPMN2-enforce UML-enforce SoaML stereotype application is followed in all QVT relations defined. Nevertheless, in some relations SoaML is also added as checkonly domain to check the existence of previously created SoaML elements needed to assure the consistence of the generation. This is the case for example in the ServicesArchitecture generation, which consists of references to SoaML elements created previously.

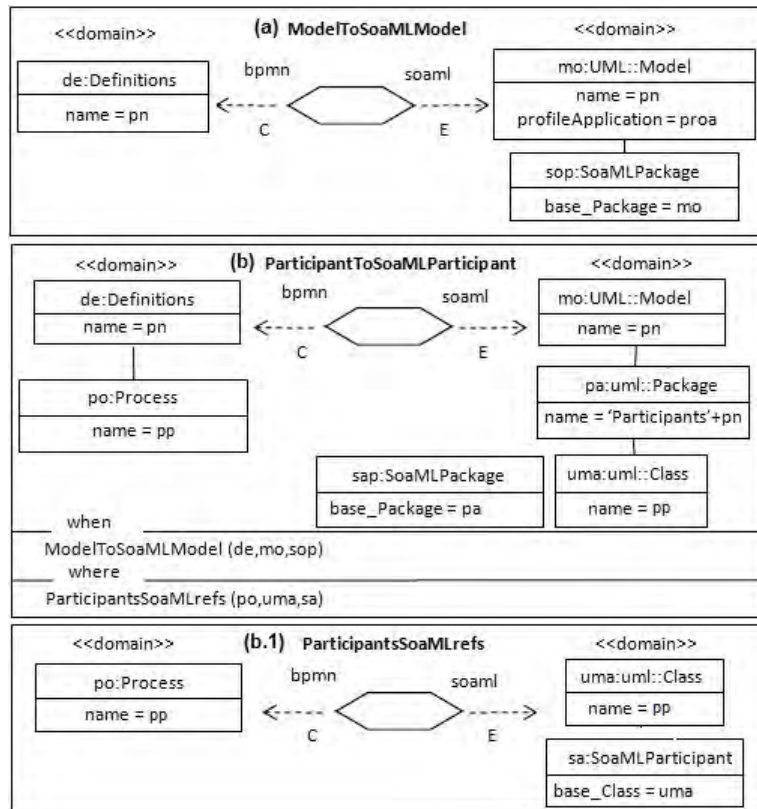
#### 8.4.2. QVT relations defined

In this section the QVT relations defined for the generation of service models from BP models are presented in QVT graphical form explaining the definitions; the corresponding code is presented in Appendix C. There are some relations that remain the same through all the generation options provided such as the generation of Participants, but there are some relations that change depending on the option chosen and on the existing elements in the BPMN2 model, such as in the case of bidirectional or unidirectional services.

In the following we present the QVT transformations defined for the bidirectional case for the options ServiceTask and ServiceTask with Interface, Operation and Message for it, as the unidirectional case is the same but without the generation of the consumer interface, and the ServiceInterface case adds another Interface referencing the ones created for the provider and consumer.

### 8.4.2.1. Model, Participants and Messages rules

From the Definitions element in the BPMN2 model, we generate the SoaML Model to hold the rest of the generated elements. In this relation we also define the application of the SoaML profile to the generated model. In Figure 8.13 (a) the top relation for the SoaML Model generation is shown in QVT graphical form. This relation remains the same through all the generation options provided.



**Figure 8.13.:** QVT relations for generating: (a) Model and (b) Participants in QVT graphical form

From each Process in the BPMN2 model we generate a SoaML Participant based on a UML Class, to which we will assign the Ports to provide or consume the generated services, after the services are generated. We create a package to hold all the generated participants in the top relation, and in the invoked relation we create the UML Class for the Participant and apply the stereotype SoaML Participant to it. In Figure 8.13 (b) the top relation and the relation invoked for the SoaML Participants generation is shown in QVT graphical form. These two relations remain the same throughout all the generation options provided.

The generation of MessageTypes differs depending on the existence of Messages elements in the BPMN2 model and the option being generated, that is bidirectional or unidirectional services. When the generation is based only on the ServiceTask identification, the SoaML MessageTypes are created from the existence of MessageFlows in the BPMN2 model where the targetRef is a ServiceTask and the sourceRef is a Task in another process.

In the top relation we generate a package to hold all the generated MessageTypes, and in the invoked relation we generate the pair of UML Classes corresponding to the service: one for the provider and the other for the consumer, and apply the MessageType stereotype to them. In Figure 8.14 (a) the top relation for the generation of MessageType from the ServiceTask is shown in QVT graphical form, for the bidirectional option, and in (a.1) the invoked relation.

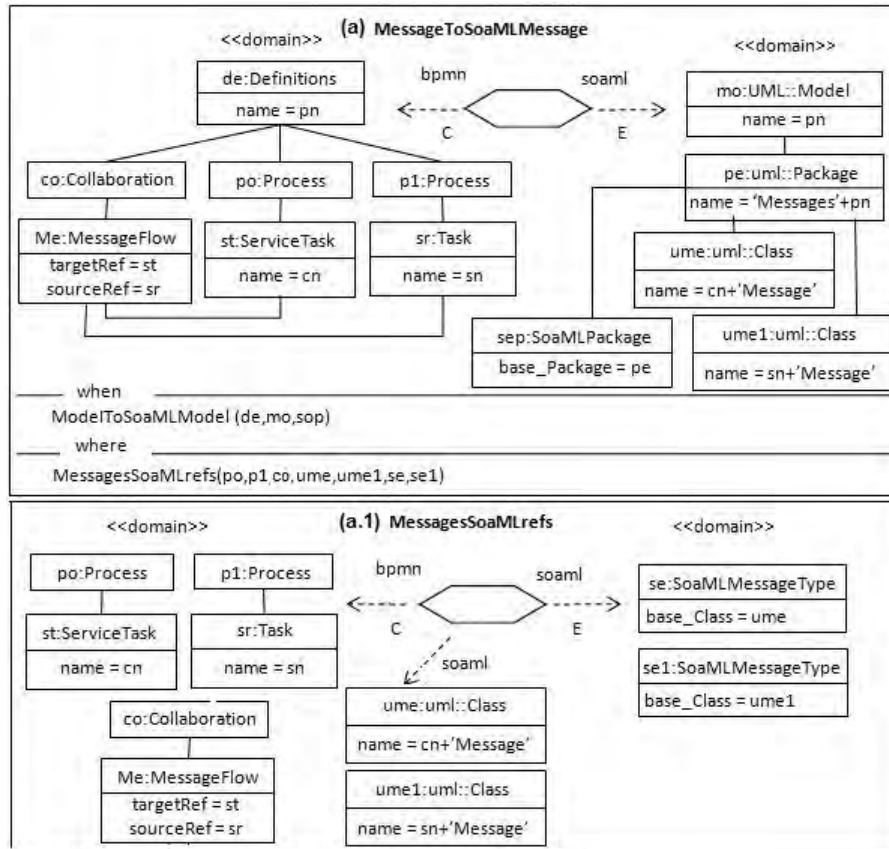
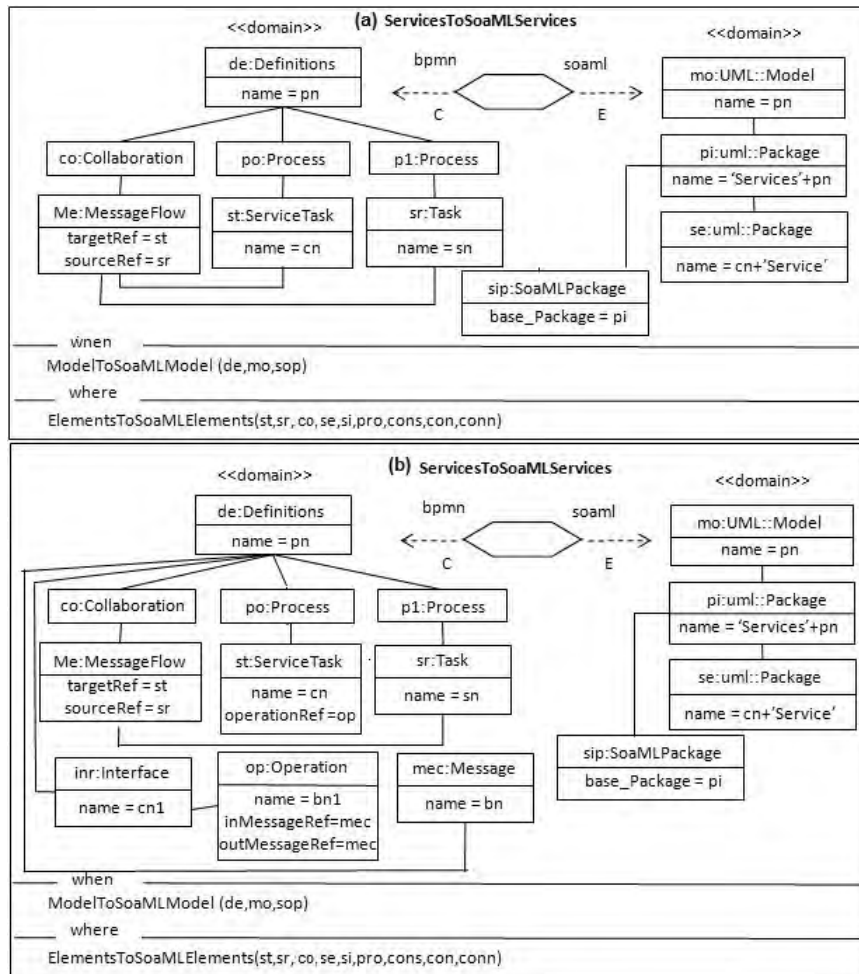


Figure 8.14.: QVT relations for generating MessageTypes in QVT graphical form

In the case where the Message for the provider Interface is specified in the BPMN2 model, the MessageType for the provider Interface is created with the same name of the BPMN2 Message. The unidirectional generation option differs in that only the Message for the provided Interface is generated. We generate MessageTypes based on UML Classes with the name of the Interface plus the word Message, to be used as parameters in the Operation defined by the service Interface for the corresponding generated service. We could also generate the rest of the options defined in the SoaML standard for the MessageTypes: DataTypes and Signal, by changing the UML elements in the transformations.

#### 8.4.2.2. Services rules

The generation of services is based on the identification of ServiceTasks in the BPMN2 model and the MessageFlow in which the ServiceTask is the targetRef and a Task in another process is the sourceRef. We identify the Task that is sending the invocation message as the consumer of the service, and the ServiceTask as the service. In the unidirectional case only one interface is generated to be the one provided by the service to be invoked, and in the bidirectional case we use the information of the sourceRef Task to generate the elements for the consumed Interface. In Figure 8.15 the QVT top relations for the generation of services are shown: (a) the top relation to generate services from the ServiceTask, and (b) the top relation to generate services from the ServiceTask with the Interface, Operation and Message defined for the service provider.



**Figure 8.15.:** QVT top relations for services generation (a) from ServiceTask and (b) with Interface, Operation and Message for the service provider, in QVT graphical form

In both relations, we first generate a package to hold all the generated services, and then for each service identified we create a package in the Services package with the name of the ServiceTask, in which we include the elements to specify it completely. The relation in the “where” clause then generates the corresponding Interfaces (provided and consumed in the bidirectional case, provided in the unidirectional case, and provided, consumed and ServiceInterface in the bidirectional ServiceInterface case), as well as the usages between the generated Interfaces (and realization and uses relations in the bidirectional ServiceInterface case), and the associated ServiceContract which defines the roles of consumer and provider based on the generated Interfaces and their dependencies. In Figure 8.16 the QVT relation for the complete generation of services from ServiceTask is shown, corresponding to the top relation (a) in Figure 8.15, for the simple UML Interfaces bidirectional case.

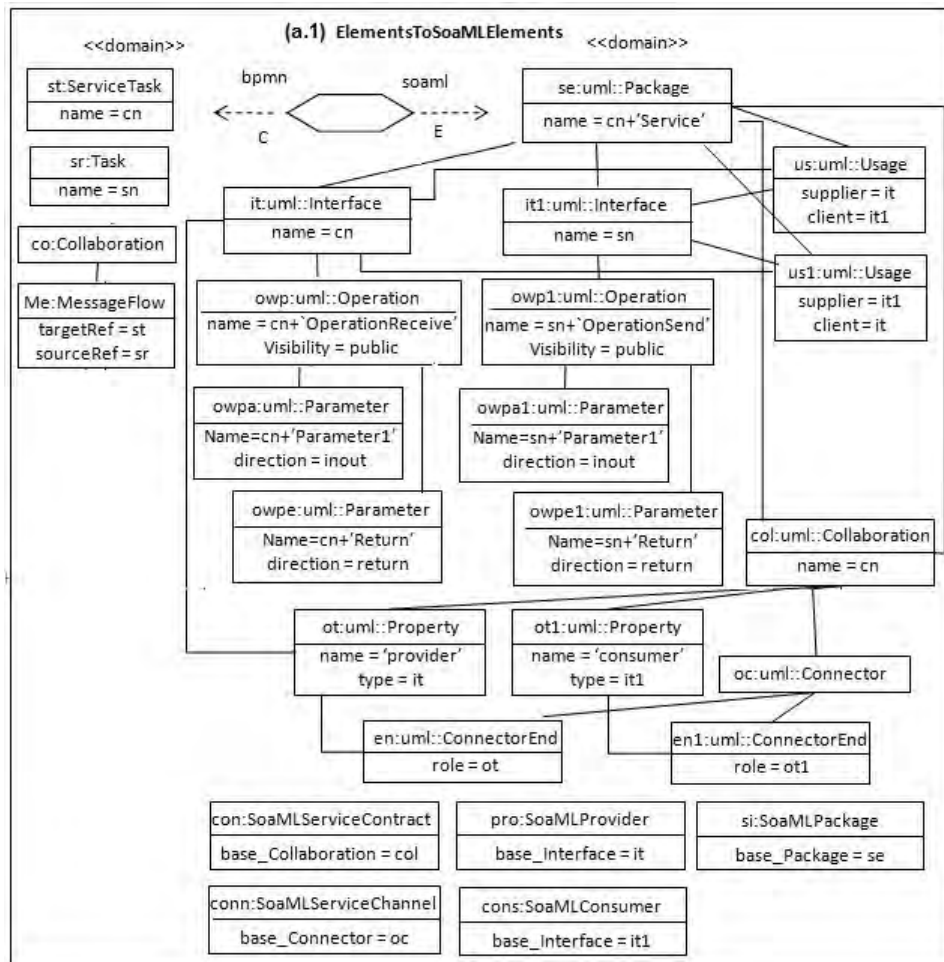


Figure 8.16.: QVT relation for services generation from ServiceTask in QVT graphical form

When the Interface, Operation and Message associated with the ServiceTask are also specified in the BPMN2 model we generate the elements for the service based on them, corresponding to the top relation (b) in Figure 8.15. The bidirectional case also generate the elements for the consumer based on the information of the sourceRef Task, as there is no way in BPMN2 to associate an operation or interface with a Task but with the name, so we decided not to ask for this information since we can generate it automatically. The unidirectional case is the same as in the previous options but only generating the provider of the service.

#### 8.4.2.3. Ports and MessageType update rules

After all the elements for the specification of the services are in place, we have to update the Participants in order to associate the Ports corresponding to the generated services to them, and the types of the parameters of the operation in the generated interface to be those of the generated MessageTypes. Figure 8.17 shows the QVT relations for the Participants Ports update, for the bidirectional option with ServiceTask marking.

The rules for Ports does this required update, adding the Ports to the Participants and applying the stereotypes Service and Request, to indicate a provided or a consumed service, respectively. We search for the created participants that correspond to the process in which the activities associated to the provider and consumer of the service belong to, and we update the base UML Classes for the Participants creating the Ports, assigning to them the type of the corresponding interface and applying the corresponding stereotype. In the unidirectional case only the provider is checked, and the type of the Request Port for the consumer is assigned as conjugated of the one corresponding to the Service Port.

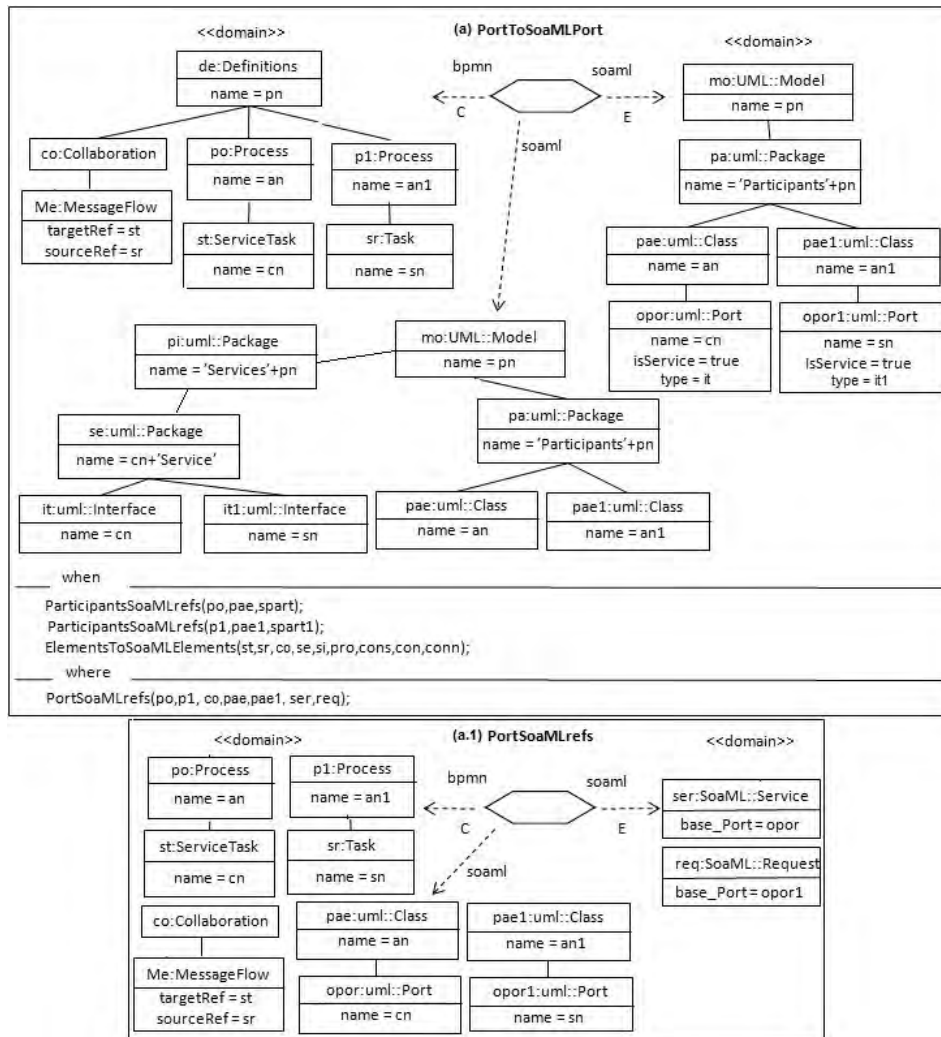


Figure 8.17.: QVT relations for Participants Ports update in QVT graphical form

To assign the type of the parameters of each created Operation in each Interface, we search for the services to which the interfaces correspond, and we update the type of the parameters assigning the MessageTypes created. The bidirectional option with Interface, Operation and parameters for the ServiceTask is similar but changing the rules for checking the BPMN2 model to verify the existence of these elements. The MessageTypes rules are not shown as conceptually they are very similar to the update of Ports but for the type of the parameters.

#### 8.4.2.4. ServicesArchitecture rules

The last rules to be executed correspond to the ones for creating the ServicesArchitecture, as this is based on all elements created previously in the SoaML model. In Figure 8.18 the QVT relations for the ServicesArchitecture generation are shown, in QVT graphical form, for the bidirectional option from ServiceTask.



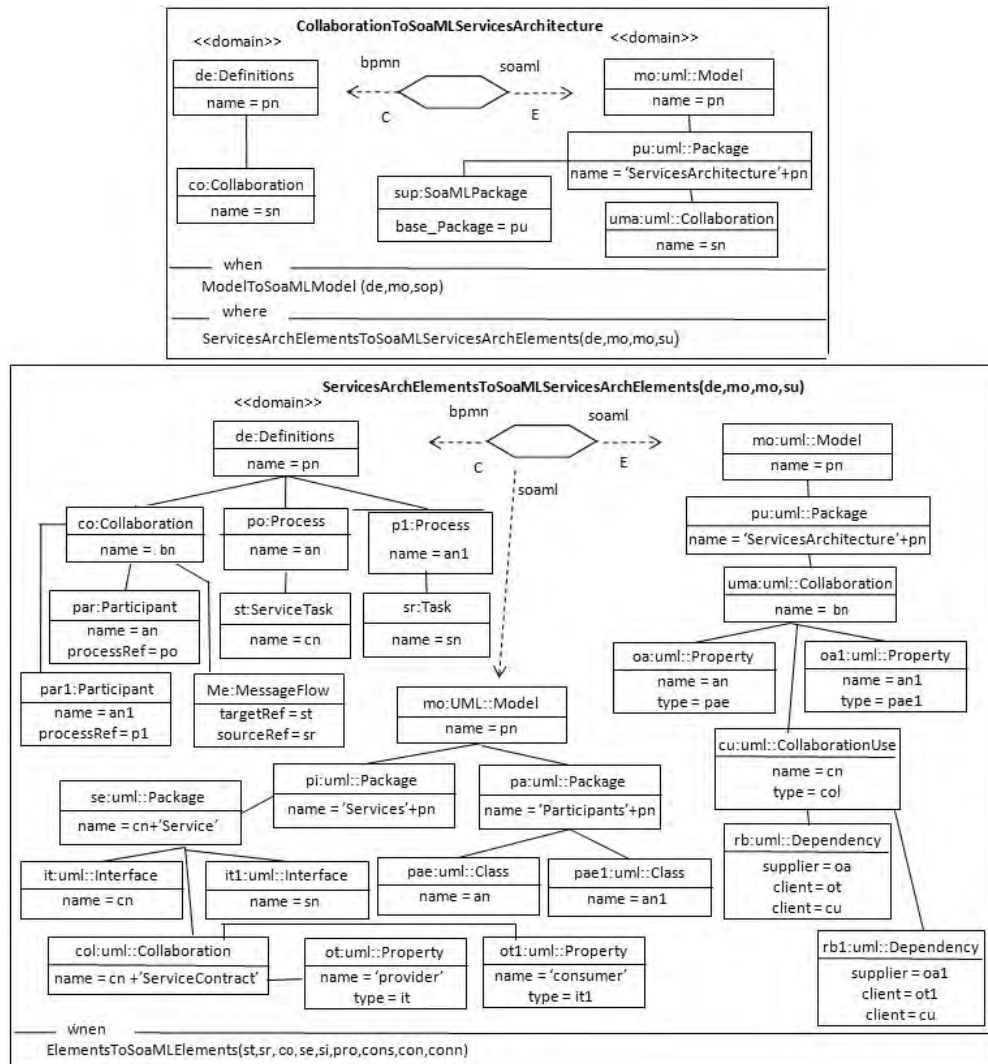


Figure 8.18.: QVT relations for the ServicesArchitecture generation in QVT graphical form

From the collaboration element in the BPMN2 model we create the ServicesArchitecture which is a UML collaboration, and inside it we generate: the participant parts referring to the generated Participants, the collaboration uses referring to the generated ServiceContracts for the services, and the role bindings between participant parts and collaboration uses according to the roles defined by the ServiceContracts.

In the top relation we generate a package to hold all the elements to be generated named ServicesArchitecture, and in the invoked relation we generate the rest of the elements mentioned before. The generated participant parts are typed with the generated Participants, the collaboration uses are typed with the ServiceContract defined roles provider and consumer, which were typed with the service interfaces when generated. The role bindings are created with one end assigned to the participant part and the other end assigned to the collaboration use that corresponds to the generated service.

In the bidirectional option with ServiceTask plus Interface, Operation and Parameter elements, the checking of the BPMN2 elements include these, and in the unidirectional option the change is that the type of the provider and consumer is that of the provider interface, conjugated for the consumer.

### 8.4.2.5. QVT transformations code

For the sake of clarity, the complete code for the QVT transformations for each of the relations presented in the previous sections, is shown in Appendix C.

### 8.4.3. Empirical evaluation

We have tested the defined QVT transformations on a set of models we have designed based on real cases from the General Hospital of Ciudad Real, and known examples such as the Travel Agency or the Buyer-Reseller processes, also used in the experiment to validate the QVT transformations, which are shown in Appendix D. In Table 8.1 the BPMN2 models evaluated are presented along with the results obtained from the generation and the execution times for the QVT transformations options defined, for the *ServiceTask* marking generation.

We have selected these models based on their complexity in terms of number *ServiceTasks* to be generated, *Participants*, and *MessageFlows* between participants in the collaborative BP. We have assessed the results of each generation to evaluate if the main objective for each one -the expected SoaML service model generated- is fulfilled, and to evaluate the times that each generation takes.

The columns presented in Table 8.1 are as follows: BPMN2 model shows the name of the BP model used as input for the generation, number of *ServiceTask* (#ST) shows the number of activities that have been identified as *ServiceTasks* for the generation, number of *Participants* (#P) shows the number of *Participants* in the collaboration, and number of *MessageFlows* (#MF) shows the number of interactions between participants, which can involve a *ServiceTask* or not.

The column SoaML service model indicates the results for the generated model, for the design options presented: *bidir1* for the bidirectional with UML simple *Interfaces*, *bidir2* for the bidirectional with *ServiceInterfaces* and *unidir* for the unidirectional with UML simple *Interfaces*. For each design option, the Time (T) is presented in milliseconds (ms), the quantity of elements generated (#E) and the quantity of set features (#F) for the generated elements.

**Table 8.1.:** SoaML service generation times for BPMN2 models of different size

BPMN2 model	#ST	#P	#MF	SoaML model					
				bidir1		bidir2		unidir	
				T (ms)	#E/ #F	T (ms)	#E/ #F	T (ms)	#E/ #F
Patient Hospital	4	3	9	25857	149 357	11171	161 389	10618	116 273
Travel Agency	11	6	11	234088	373 931	70661	406 1019	70645	284 700
Driving license	7	5	13	162312	248 608	42041	269 664	42330	191 461
Grant Loan	4	3	7	28738	149 357	11158	161 389	10998	116 273
Submit Article	10	4	10	90875	330 838	33655	360 918	33385	249 628
Buyer - Reseller	4	3	5	2631	149 357	1167	161 389	1088	116 273
Shopping cart	7	4	8	34723	241 601	11869	262 657	12052	184 454

In Figure 8.19 (a) a summary of the number of *ServiceTasks*, *Participants* and *MessageFlows* in the BPMN2 model is presented graphically, and in (b) the number of generated elements and set features in the generated SoaML service model are shown, which will be the basis for the discussion about the results obtained.

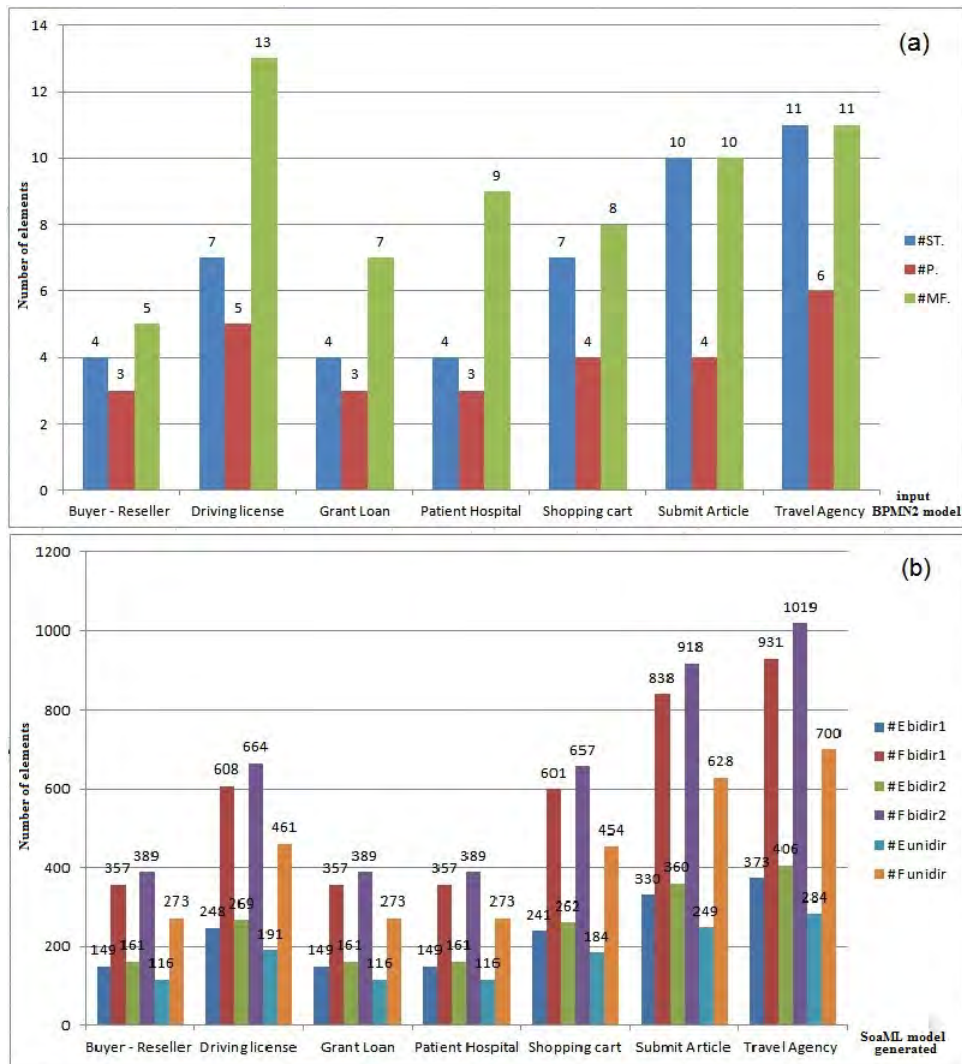


Figure 8.19.: Elements in BPMN2 and SoaML models

The times for the generation of bidirectional UML simple *Interfaces* services (bidir1), are below one minute for the majority of the BPs, a minute and a half for the Conference submission -which has ten *ServiceTasks*-, almost three minutes for the Driving license -which has seven *ServiceTasks* but more *Participants* and *MessagesFlows* than the previous one-, and almost four minutes which is the maximum generation time for the Travel Agency -which has eleven *ServiceTasks* and the maximum number of *Participants* which is six-, in which almost four hundred elements are created and near a thousand features are set.

When the number of *ServiceTasks* and *Participants* for two BPMN2 models is the same, the same number of elements is created in the two generated SoaML models, which is the expected behavior for the generation. The times are increased by the number of *Participants* and *MessagesFlows* between them, as the definition of *Service* based on the *MessagesFlows* between all *Participants* has to be checked for all *MessageFlows*, and the rule to update *Participants* to add the corresponding *Ports*, also checks all the pairs involved to assign the created *Services* to the *Participants*. This can be seen by comparing the Shopping cart and Driving License BPs generation times, in which both have the same number of *ServiceTask* but differ in the number of *Participants* and *MessagesFlows*.

For the bidirectional *ServiceInterface* (bidir2) and unidirectional UML simple *Interfaces* (unidir) generation the times are below the previous ones, although in some cases the number of generated elements and features set are greater. This is due to the fact that for the *ServiceInterface* the types assigned for the consumer and provider are the same -the *ServiceInterface* and its conjugated- so the update rules and assignments have fewer elements to check on the already created SoaML

elements in the target model being generated. In the unidirectional case it is the expected behavior as only the provider elements are being created so the elements to be checked in the target model being generated are fewer than in both bidirectional generations. In Figure 8.20 the times for the generation discussed are presented.

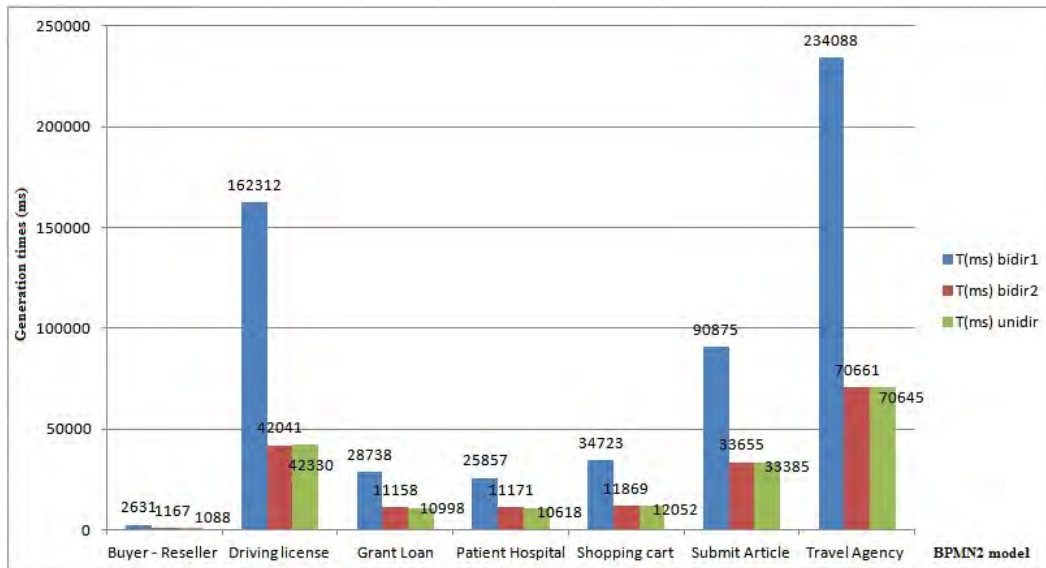


Figure 8.20.: Generation times for BPMN2 model and generation option

To check that the SoaML service model was generated correctly, the resulting SoaML XMI file was loaded into the environment with the EMF editor, analyzing the generated elements, and then it was loaded into the Eclipse SoaML plug-in to visualize the models graphically. Also, the QVT transformations defined were validated by means of an experiment carried out to assess the suitability of the QVT transformations defined, and the understandability of the SoaML service models generated. The design, execution and results from the experiment are presented in chapter 10.

## 8.5. Example of application

To show the application of our proposal we use the same “Patient MAS” BP adapted from the Hospital General de Ciudad Real (HGCR) that we presented in chapter 6. Nevertheless we show the BPMN2 model in Figure 8.21 again so the correspondence of elements for the generation can be seen easily.

The tool support for carrying out the generation process in MINERVA was introduced briefly in section 8.2 and is described in chapter 9. That being so, we will not repeat it here. The different steps that are mentioned in the following are performed with the defined tools.

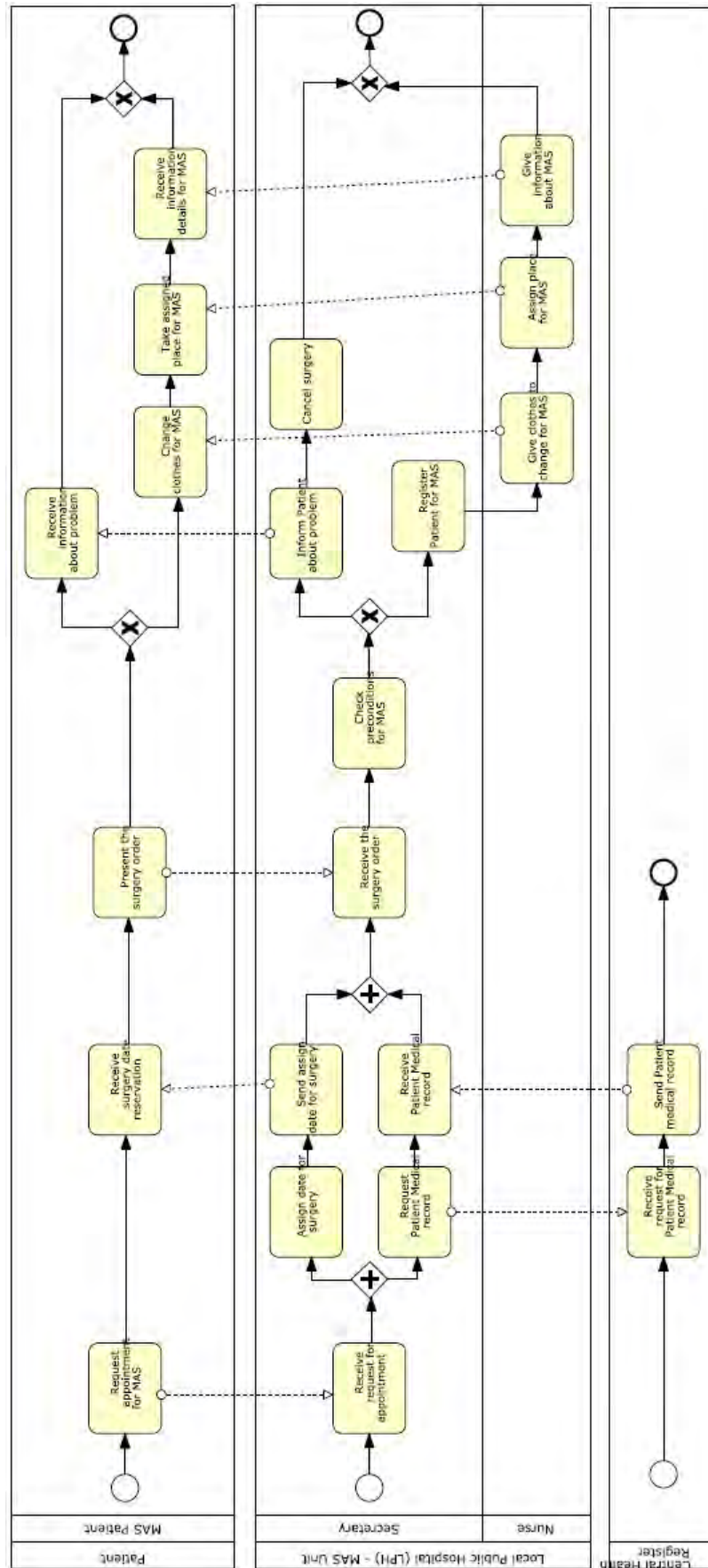


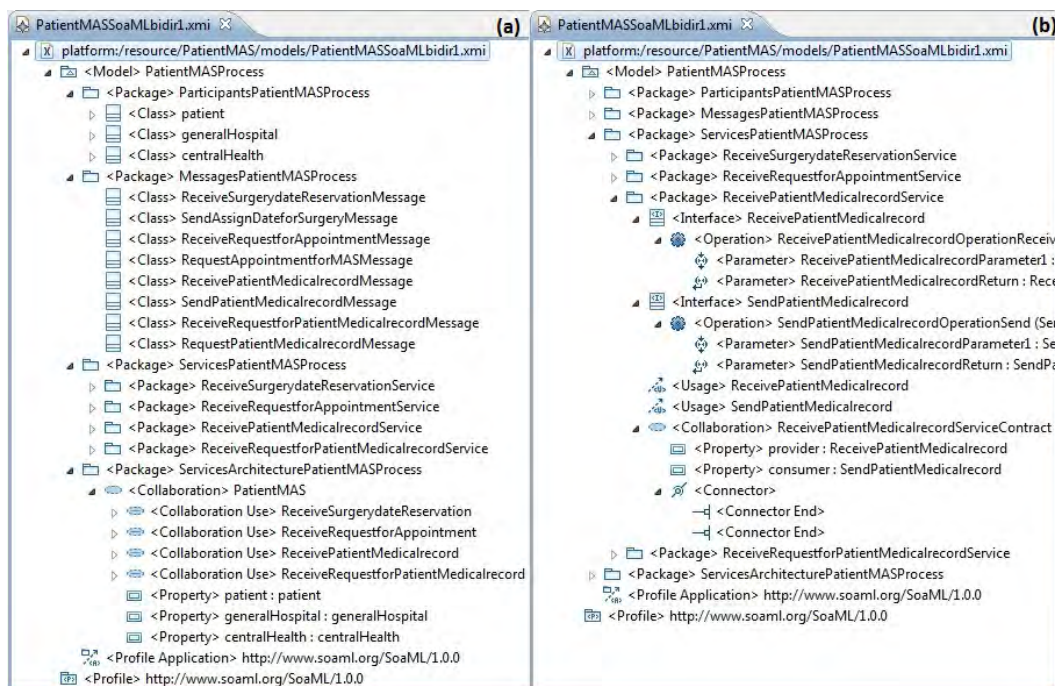
Figure 8.21.: Patient MAS from the HGCR in BPMN2

### 8.5.1. Application of the generation procedure

After the BPMN2 model is created and saved in .BPMN2 XML format, it is loaded in the Eclipse MINERVA design environment to be used as the input for the transformations. The Service Architect then identifies the needed Services and sets the ServiceTask type for the following activities: “Receive Request for Appointment”, “Receive Request for Patient Medical record”, “Receive surgery date reservation” and “Receive Patient Medical record”, after agreeing with the Central Health Register on the interaction to be performed. The patient side task for receiving the date assigned for the surgery will be implemented accessing the email or cell phone number provided by the client so the Hospital has its record with the required information. Once the ServiceTasks are identified in the BPMN2 model, the XMI file is generated with the XSLT transformation, to be used as the input for the QVT transformations.

Once the BPMN2 model with ServiceTasks in XMI format is in place, the QVT transformations can be executed to generate the corresponding SoaML service model, also in XMI format. We choose to execute the bidirectional case with UML simple Interfaces (bidir1) as it generates several elements in the SoaML service model. As mentioned before, the unidirectional case only generates the interface for the provider, and the bidirectional case with ServiceInterface is similar to the bidirectional with UML simple Interfaces but adding the ServiceInterface to type the Ports, which makes the resulting model more difficult to understand.

In Figure 8.22 the generated SoaML XMI file for the services model is shown in the Eclipse ecere editor, in (a) the general structure showing the defined packages and classes for Participants and Messages, packages for each generated service and elements in the ServicesArchitecture, (b) the complete specification of the service “Receive Patient Medical record” with its generated interfaces (provider and consumer), operations and parameters for each one (of type generated MessagesTypes), usages between interfaces and the associated ServiceContract defining the provider and consumer typed with the generated interfaces and their connection.



**Figure 8.22.:** Generated SoaML XMI file in Eclipse ecere editor showing the UML base model: (a) SoaML model general structure, (b) Service specification

In Figure 8.23 the SoaML stereotypes that are applied to the base UML model generated are shown, (a) in the Eclipse ecere editor and (b) in XML view.



**Figure 8.23.:** Generated SoaML XMI file stereotypes application (a) Eclipse ecere editor (b) XML view

The first Package in Figure 8.23 (a) corresponds to the Model element generated, then for each generated Package the application of corresponding stereotypes: Participants, MessageTypes and for each Service the: Provider, Consumer, ServiceContract and ServiceChannel. After this, the Service and Request port stereotype assignment to ports in Participants, and finally the ServiceArchitecture. At the end of the file the .ecore metamodels referenced are shown. In (b) the XML view shows the references to the generated UML base elements, by means of the absolute path from the root element -the package corresponding to the Model-, shown on the top of the figure as “/0”.

The SoaML XMI file generated is imported in the Eclipse SoaML plug-in developed so as to visualize the generated SoaML service model, which is described in chapter 9.

## 8.5.2. Generated SoaML diagrams

In this section the SoaML diagrams corresponding to the generated SoaML model are presented, we ease the identification of diagrams by the definition of the package structure we generate, so the ServiceArchitecture diagram goes into the ServiceArchitecture package, the same occurs for Participants and Messages, and for each Service, both the associated Interfaces diagram and the ServiceContract diagram go into the package corresponding to the service.

### 8.5.2.1. ServiceArchitecture diagram

The ServiceArchitecture diagram shows the three generated Participants: Patient, generalHospital and centralHealth, and the reference to the ServiceContracts for the generated services: “ReceivePatientMedicalrecord”, “ReceiveRequestforPatientMedicalrecord”, “ReceiveSurgerydateReservation” and “ReceiveRequestforAppointment”. The roles each participant play within each service are also shown in the role bindings that link each participant with the services in which they participate. In Figure 8.24 the ServiceArchitecture diagram generated is shown, which is the same for all generation options.

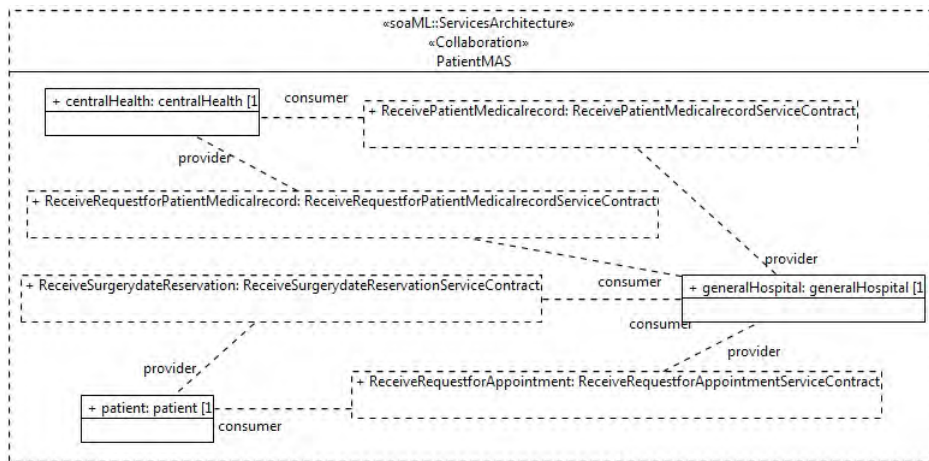


Figure 8.24.: Generated ServicesArchitecture diagram

### 8.5.2.2. Services Interfaces and ServiceContracts

For each service identified the generated service specification includes the diagrams for: the service Interfaces (one for the provider and another for the consumer) with their associated usages relations, the Operation and Parameters generated to interact with the service, and the ServiceContract specifying the provider and consumer roles to interact with the service typed by the generated interfaces. The differences between the generation options are in the generation of the Interfaces and ServiceContracts associated to the services, so in order to show what can be obtained by executing each of the transformations, we will present all three of them.

#### Bidirectional with simple UML Interfaces generation

The bidirectional case shows the generation of the two interfaces: one for the provider of the service and another for the consumer. The usages between them implies that the consumer uses the provider interface to invoke the service, and the provider uses the consumer interface to send the answer as defined by the service specification. The corresponding ServiceContract shows the consumer and provider roles typed by the consumer and provider interfaces respectively. Figure 8.25 shows an example of the specification for the generated service “ReceivePatientMedicalrecord” based on ServiceTask identification for the bidirectional UML simple Interfaces generation.

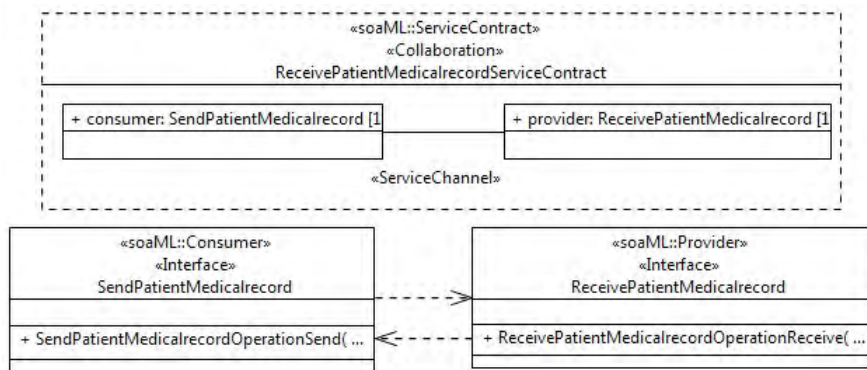


Figure 8.25.: Generated services specification bidirectional option

#### Unidirectional with simple UML Interfaces generation

In the unidirectional case only the provider Interface is generated, without generating one for the consumer which is not typed in the associated ServiceContract. Later in the Ports specification the type of the Request Port corresponding to the consumer will be typed as the conjugated of the provided interface. Figure 8.26 shows the specification for the service generated “ReceivePatientMedicalrecord” based on ServiceTask identification for the unidirectional UML simple Interfaces generation.



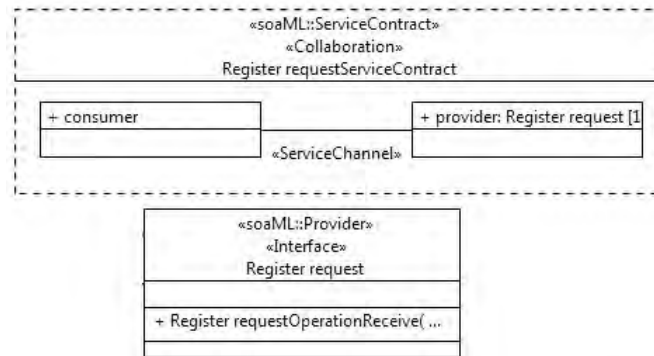


Figure 8.26.: Generated services specification unidirectional option

### Bidirectional with ServiceInterface generation

In the ServiceInterface case three interfaces are generated: the UML simple interfaces as before for the provider and consumer of the service, and the ServiceInterface one which realizes the provider interface and uses the consumer interface. In the corresponding ServiceContract the types for both the consumer and provider are these of the ServiceInterface, as it defines which is the interface provided and which is the consumed one, via the realization and uses relationships.

Figure 8.27 shows the specification for the generated service “ReceivePatientMedicalrecord” based on ServiceTask identification for the bidirectional option with ServiceInterface.

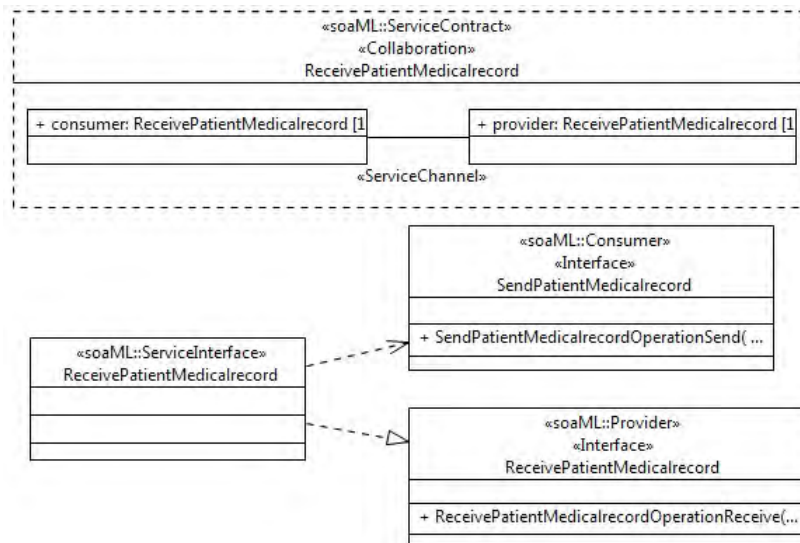


Figure 8.27.: Generated services specification ServiceInterface bidirectional option

#### 8.5.2.3. Participants

The generated Participants are the same for all the options provided as mentioned, what changes is the type assigned to their ports, which depends on the type of generation, as shown before. In the bidirectional with simple UML Interfaces case the Service Port is typed with the provider Interface as defined in the Service specification, and the associated Request Port of the consumer is typed with the corresponding consumer Interface. In Figure 8.28 the generated Participants with corresponding Service and Request Ports are shown for the bidirectional option with UML simple Interfaces.

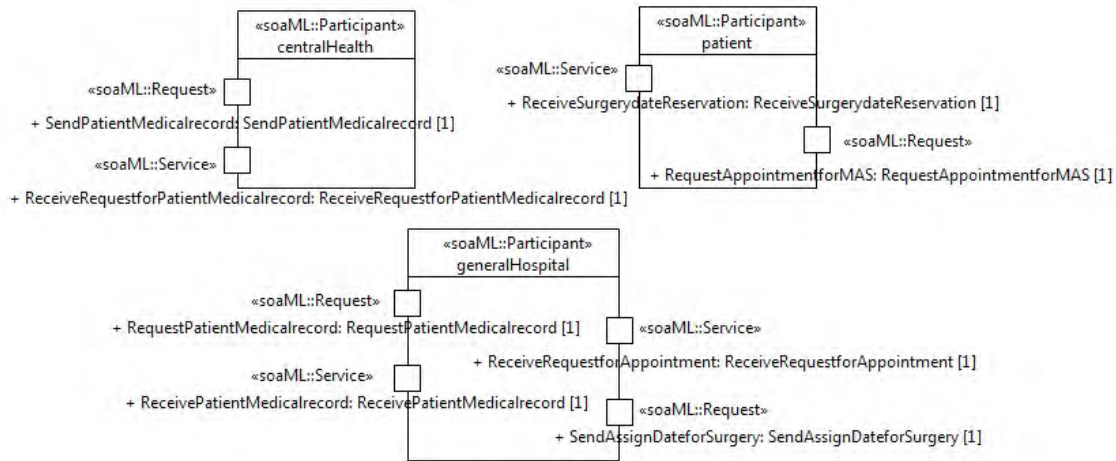


Figure 8.28.: Generated Participants and Ports bidirectional option

We do not present the other two generation cases here, as the generated Participants are the same, the only changes are located in the associated ports: in the unidirectional case the Service Port has the type of the provider interface, and the Request Port its conjugated, and in the ServiceInterface case Service and Request Ports both have the type of the ServiceInterface, as it defines which Interface realizes -that corresponds to the provider-, and which Interface uses -that corresponds to the consumer-.

#### 8.5.2.4. MessageTypes

The generated MessageTypes which are the type of the parameters of the operations in the interfaces for each service, are generated one for each operation and used both as type for the input and the return parameter. As mentioned MessageTypes are generated as a Class with no attributes, which have to be added to the generated design manually by the Architect or Developer. In Figure 8.29 the generated MessageTypes for the ServiceTask identification in the bidirectional option with UML simple Interfaces are shown. For the bidirectional option with ServiceInterface the MessageTypes generated are the same, and in the unidirectional option we only generate the MessageTypes for the operation of the provider interface.



Figure 8.29.: Generated MessageTypes bidirectional option

In the case where the Interfaces, Operations and Messages are also defined in the BPMN2 model, the only difference in the generation of all the diagrams presented previously, is that the generated names for these elements are those of the original Interfaces, Operations and Messages in the BPMN2 model.

## 8.6. Conclusions

In this Chapter the model-driven approach of MINERVA has been presented, which is an MDA application based on the BPMN2, QVT, MOF, XMI, SoaML and UML standards and metamodels. The definition of the correspondences between the two metamodels BPMN2 and SoaML were presented, along with the definition of the QVT transformations and corresponding rules based on the defined correspondences. The generation of SoaML service models from BPMN2 models is integrated in BPSOM as presented in chapter 7.

Three main transformations are provided to generate different SoaML service model designs, based on the communication patterns bidirectional with UML simple Interfaces, bidirectional with ServiceInterfaces and unidirectional with UML simple Interfaces. In all three cases the generation is based on the identification of ServiceTasks in the BPMN2 model, providing the generation of different service specifications depending on the option chosen. Three other optional transformations are provided when other elements are present in the BPMN2 such as Interfaces, Operations and Messages for the provider of the service, in which the names of these elements are preserved in the generation. For each defined rule its definition was presented in QVT graphical syntax, and the complete code can be seen in Appendix C.

Finally an example was provided to show the application of the defined procedure for the generation and the results of the generated SoaML service model, for each of the QVT transformations based on ServiceTask identification defined. The QVT transformations defined were validated by means of an experiment which is presented in chapter 10.



## Chapter 9.

### Tools support

This Chapter describes the tools support defined for MINERVA framework which includes tools for modeling and executing BPs, a main integrated environment for developing service-oriented systems from BPs with a model-driven approach, and a proof of concept tool for calculating and visualizing the execution measures of BPEMM, to analyze BPs execution.

The Chapter is organized as follows: in section 9.1 a description of the tool support defined in MINERVA is presented, which is further detailed for each of the BPCIP lifecycle phases in section 9.2 and also for the guidance through the method of work, and finally in section 9.3 conclusions for the chapter are discussed.

The contents of this chapter are related with all previous chapters in which the tools support is applied, and with chapter 10 which presents the validation of MINERVA framework and use of the tools support presented here.

#### 9.1. Introduction

The availability of tools is an essential aspect for the use and success of any method or technique. The provision and use of tools is defined in the tool dimension of MINERVA, and its main objective is to provide support for performing all the activities defined in the MINERVA method of work through the BPCIP lifecycle, from BP modeling through its service-oriented development to its execution and evaluation.

A key goal when defining the support of tools for each phase was to provide an integrated chain of development, where artifacts developed in one tool as an output of a certain activity could also be used as the input for the tool used in the next activity in the defined lifecycle. The BPCIP phases provide the basis for the definition of tools:

- **Design&Analysis:** a tool for modeling BP which provides functionalities such as saving the BP model in BPMN2 format and exporting it in a human readable way as an image (jpg, pdf, etc.), that can be used by the business and the software area.
- **Configuration:** an integrated tool for the development of service systems from BP. It allows the models in BPMN2 format to be used as input and provides the chain of development from modeling to design with SoaML (including support for the automatic generation) and implementation in the selected technology, to be used by the software area.
- **Enactment:** a tool that provides a process engine to execute BP models in one of the main languages for execution: XPDL, BPEL or BPMN2, registering the data needed from the BP cases execution, to be used by the business and the software area.
- **Evaluation:** a tool that allows the analysis of the execution of BPs based on the data registered, including the calculation of the BPEMM execution measures to analyze the execution to find improvement opportunities, that can be used by the business and the software area.

The tools to be selected should have as much as possible a free license or if not at least be freeware, as defined by GNU<sup>1</sup>; this will allow the tools to be used in all kinds of organizations, and also enable

---

<sup>1</sup><http://www.gnu.org/philosophy/categories.html>

us to adapt them if needed. The set of tools to provide support for the method of work defined for MINERVA as presented in chapter 4 is shown in Figure 9.1 to give the reader the context of the definitions that are described below.

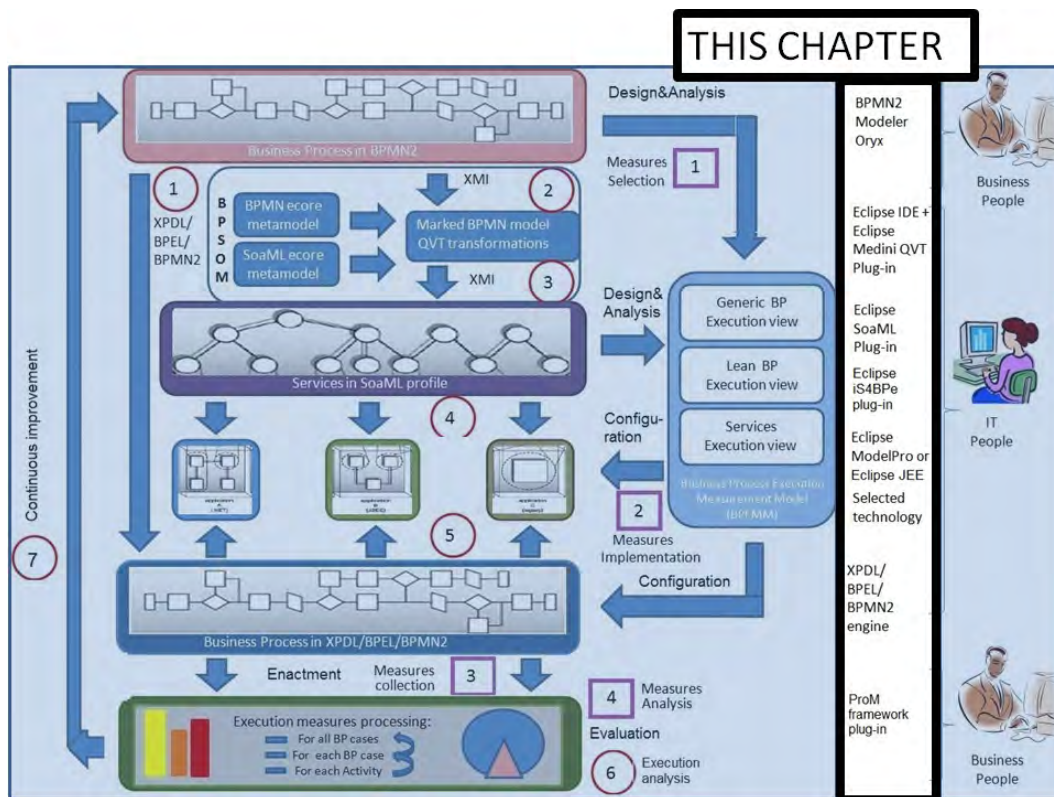


Figure 9.1.: MINERVA method of work through BPCIP

## 9.2. BPCIP phases tools support

The tools support is presented in this section for each BPCIP phase to provide the complete vision of the method of work of MINERVA. For each of the BPCIP phases the corresponding tools support has been defined, which makes it possible to perform the activities defined producing the defined artifacts. The approach we have followed was to reuse existing tools as much as possible providing they support the functionalities needed by MINERVA definitions. To do so we have performed evaluations of several existing tools, and others have been chosen because of specific capabilities and our previous experience using them.

### 9.2.1. Tools decision criteria

The selection of tools for BP modeling and execution corresponding to the Design&Analysis and Enactment phases respectively, is based on a general procedure for performing evaluation of BPMS which was defined in the context of the project with the Hospital General de Ciudad Real (HGCR, Ciudad Real General Hospital), Spain, in February 2009.

It includes several characteristics based on an adaptation of the TEC-Technology Evaluation Center<sup>2</sup> list, and two groups of characteristics: the first one to be applied, group A, defines characteristics which if not provided, cause the tool to be removed from the selection process; the second one, group B, defines characteristics that allow the tool to be evaluated in depth, having passed group A.

<sup>2</sup><http://www.technologyevaluation.com/>

The document also includes the definition of two types of scales to assess each characteristic: a first scale defining six values and the associated labels, and a second one used to weight the importance of the characteristic in the whole set, both shown in Table 9.1. The second scale is defined so the appropriate stakeholders (business, software) can select the importance for the characteristics that they want most.

**Table 9.1.:** Scales for assessing BPMS characteristics

Scale 1 - to assess each characteristic		
Value	Label	Description
5	FULL-SUPP	Fully supported
4	PART-SUPP	Partially supported
3	MOD-SUPP	Supported via modifications
2	THIR-SUPP	Supported via third party solutions
1	CUST-SUPP	Supported via customization
0	NSUPP	Not supported
Scale 2 - to define the weight of each characteristic		
Value	Importance	Description
5	Mandatory	The tool must provide the characteristic
	Priority defined	The tool provides the characteristic in the defined rank
4	Very important	
3	Important	
2	Good to have	
1	Not important	
0	Not necessary	

As the document defines characteristics for evaluating complete BPMS, it includes specific ones for evaluating BP modelers and specific ones for evaluating BP engines. For the Configuration and Evaluation phases we have based the selection of tools on the provision of capabilities for extension, for example in the form of plug-ins, which allow the definitions in MINERVA to be integrated; we have also taken our previous experience using them as a basis for the choice of tools for these phases.

## 9.2.2. Design&Analysis phase

The business area and people, are of great importance for the MINERVA framework, as the business knowledge of the organization resides in them. The business and the software area must work together if the BPs in the organization are to be managed and improved to provide the required outputs as defined. Business Process modeling is one of the key activities in the method of work defined in MINERVA, as BP models provide the basis for the explicit view of the business, they also constitutes the main input for the service-oriented development driven by models, and set the basis for the execution of BPs in the organization by means of a BPMS. In this phase the selection of the BPEMM measures by the business people also has to be done, to set the basis for the implementation, collection and evaluation of execution measures from BPs execution. In the following the evaluation and selection of BP modelers and the selected tool for specifying BPEMM measures are described.

### 9.2.2.1. BP modeling

Although BP modeling is carried out by the business and software areas in conjunction, it is desirable for business people to be able to specify the BP models by themselves, in a user friendly and easy to use tool which provides support for the BPMN standard. The first evaluation of BP modelers was carried out by a student in a postgraduate course, by means of the following procedure:

1. study of the general characteristics of the BP modelers based on the online documentation provided and/or available tutorials and white papers.
2. installation of the product, registering any problems detected and/or lack of guidance in the installation process.
3. development of a case study using each tool installed, by modeling the “Voting BP” as defined in the BPMN v1.2 standard.
4. assessment of a set of relevant characteristics defined for BPMN modelers, which are a sub-set of the characteristics defined for the HGCR. Some of these are shown in Table 9.2.

**Table 9.2.:** BPMN modelers key user oriented characteristics

<b>1</b>	<b>BP modelling</b>
<b>1.1</b>	<b>Graphic designer</b>
1.1.1	The process flow can be graphically designed
1.1.2	High level state diagram (for business users)
1.1.3	Technical detailed diagram (for software users)
1.1.4	Drag and drop of process modelling elements
1.1.5	Requires an IDE from a third party
1.1.6	The design is stored in an structured repository
<b>3</b>	<b>BP collaboration</b>
<b>3.1</b>	<b>Check-in/Check-out</b>
3.1.1	BP designers can perform check-in/ check-out of documents
3.1.3	BP designers can have a view over all the BP versions
3.1.4	BP designers can download a particular version of the BP
3.1.7	Author’s name, group, role, date and hour are registered for each BP version
<b>3.4</b>	<b>Import/Export formats</b>
3.4.1	Import/Export in XPD L is provided
3.4.2	Import/Export in BPEL is provided
3.4.3	Other Import/Export formats are provided (i.e. pdf, jpg)

By the time the first evaluation was carried out the current version of the BPMN standard was the v1.2, and the selected tools to be evaluated were:

- BPMN modelers: Eclipse BPMN<sup>3</sup> modeler plug-in, ADONIS<sup>4</sup> community edition, Bizagi<sup>5</sup> Process modeler, Magic Draw<sup>6</sup> and BP Visual Architect<sup>7</sup>.

which were selected on the basis of their license and availability, and the last two because we had academic licenses. Other products such as the current ARIS Express<sup>8</sup> community edition, did not provide a free BPMN modeler at that time (only EPC). Please bear in mind that most of these tools now provide BPMN2 support and that the previous versions, the ones we evaluated, are no longer available in the referenced sites.

As a result of the evaluation, two BP modelers were selected for MINERVA: one for the business area, the Bizagi Process modeler, and one for the software area, the Eclipse BPMN modeler plug-in, to be integrated into the Eclipse MINERVA design environment:

- Bizagi Process modeler which is freeware, provides most of the desired characteristics, including the most important ones regarding the Graphical designer and the Import/Export formats. It provides the implementation of the complete BPMN standard in a user friendly environment which is also multilingual. At the time the evaluation was carried out no standard format was defined for the interchange of BPMN models, so XPD L was mostly used, which Bizagi provides facilities to export to. It also provides facilities to export the BP model, among other formats, as pdf and jpg.

<sup>3</sup><http://archive.eclipse.org/soa/archives/bpmn.tgz>

<sup>4</sup><http://www.adonis-community.com/>

<sup>5</sup>[www.bizagi.com/modeler/](http://www.bizagi.com/modeler/)

<sup>6</sup><https://www.magicdraw.com/>

<sup>7</sup><http://www.visual-paradigm.com/product/bpva/>

<sup>8</sup><http://www.ariscommunity.com/aris-express>



- Eclipse BPMN modeler plug-in provides an implementation of a core set of BPMN elements, very reduced, but accompanied with a BPMN metamodel compatible with the BPMN metamodel (a sub-set) released by OMG. We integrated it into the Eclipse MINERVA design so as to be able to define an initial set of transformations between the BPMN metamodel and the SoaML b1 metamodel. Although several elements needed -such as the types for the activities- were missing, it allowed us performing a proof of concept for defining the transformations.

When the BPMN2 standard was released -on January 2011- we updated the definitions in MINERVA framework including the tool support, as it presents many changes from its previous versions, such as the definition of a standard exchange format for BPMN2 models and the semantics for execution, as presented in chapter 3. Although Bizagi announced the release of the BPMN2 version for mid 2011, the release still uses BPMN v1.2 and the BPMN2 support was delayed until the end of the year. In addition, a project was started at Eclipse to update the Eclipse BPMN modeler plug-in to BPMN2 modeler, which was released by the end of the year.

As we needed a BPMN2 modeler at the time the standard was released, a brief search and evaluation of other available tools providing BPMN2 support was carried out, adding as a key feature the saving of BP models in the BPMN2 standard exchange format. Although for the BPMN2 version several tools were updating their BP modelers, when we carried out the new evaluation for BPMN2, few had been released. Because of these considerations, the Oryx editor <sup>9</sup> was selected, which provides several of the characteristics defined. The most important features were the saving of BP models in BPMN2 format and its being open source, also adding several other characteristics such as its being web based and providing support for exporting BP models in several formats such as XPD and pdf. What is more, Oryx can be used by both business and software areas, being not only user friendly and easy to use but also providing the technical requirements regarding the BPMN2 format.

Several tools and projects started providing support not only for modeling in BPMN2 but also for executing BPMN2 models directly in BPMN2 process engines based on the semantics defined in the standard, two other BPMN2 modelers can also be used which are part of BPMN2 process engines -Activiti and jBPM5- which provides similar characteristics to Oryx as they share the same core code: the Activiti Modeler <sup>10</sup> -which is related to Oryx by the Signavio commercial tool- and the jBPM5 Designer <sup>11</sup> -which was developed based on Oryx code-.

#### 9.2.2.2. BPEMM measures specification

We have described in chapter 6 that the specification of the execution measures is done by means of the GQM in textual form following the definitions of the SMO. This way of specification could be clear enough for business people to understand the BPEMM execution measures defined, and to select the ones that are more appropriate for the execution of each BP in the organization. But, following the adage “a picture is worth a thousand words”, we also think that a graphical support should be provided in order to give a quick global view of the elements involved in the measurement.

For this reason we have integrated the SMTTool [Mora et al., 2008] that implements the concepts and relationships defined in the SMO, providing a graphical view of any measurement model, using the Software Measurement Modeling Language (SMML) [Mora et al., 2011]. In Figure 9.2 an example of using SMTTool/SMML for some execution measures defined in the Generic BP view for the time dimension is shown. Several concepts and relationships are shown for the definition of the execution measures: in the first place on the left upper corner of the diagram the information need “To know the Throughput Time (TT) of BP execution” is shown, which is related with the measurable concept “Throughput Time” to its right, and below the attribute to be measured that is also the “Throughput Time (TT)” of the entity “Business Process (BP)” to its left.

---

<sup>9</sup><http://bpt.hpi.uni-potsdam.de/Oryx/Research>

<sup>10</sup><http://www.activiti.org/components.html> / <http://ge.tt/9qUwL98/v/0>

<sup>11</sup><http://docs.jboss.org/jbpm/v5.1/userguide/ch10.html>

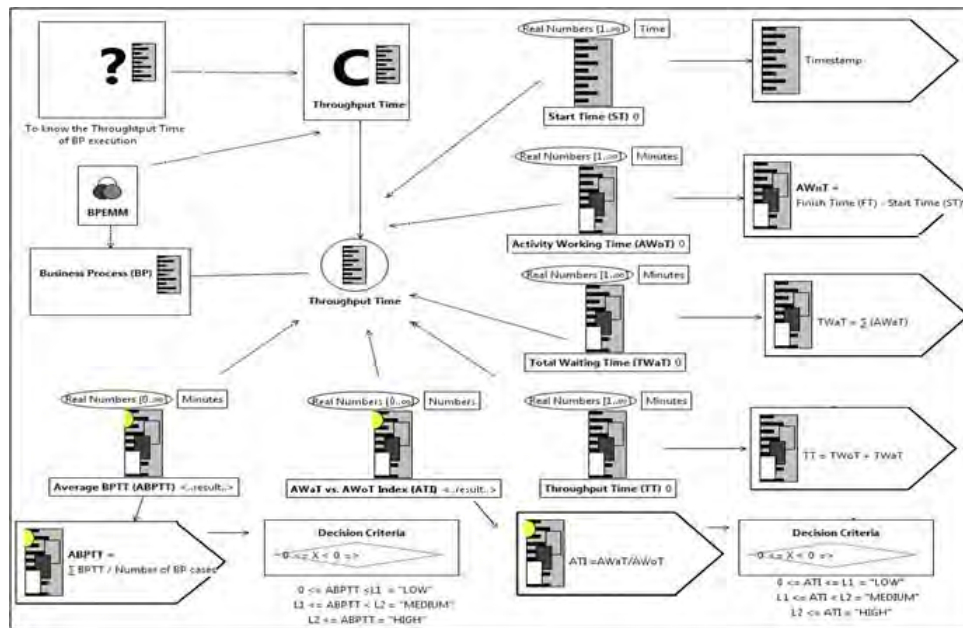


Figure 9.2.: SMTool example of BPEMM execution measures specification in graphical form

On the right and bottom of Figure 9.2 the three types of execution measures defined to measure the TT are shown, that is: base measures (identified by the rule), for example “Start Time” in the upper right corner of the diagram, derived measures below (identified by the rule with rectangles inside), for example “Activity Working Time (AWoT)”, “Total Waiting Time (TWaT)” and “Throughput Time (TT)”, and the indicators below to the left (identified by the rule with rectangles inside and the yellow lamp), for example “AWaT vs. AWAoT Index (ATI)” and “Average BPTT (ABPTT)”, each with its associated measurement approach.

The SMTool/SMML diagrams are used in the Design&Analysis phase to ease the selection of the BPEMM measures by the business people. Although the tool provides the facility for the automatic calculation of the values of any measure defined for any domain if the measures are defined based on elements of the model, here this facility is not applicable since execution measures do not measure inherent properties of the models, but of the BP cases, that is the instances of the BPs execution.

### 9.2.3. Configuration phase

The software team carrying out the implementation of the services from the BP models needs an integrate tool to be able to perform the activities defined by BPSOM without having to alternate between different tools, which can add difficulty to the process. Although the BP modeling is performed using the Oryx editor as mentioned before, once the selected activities in the BP model have been marked as of “Service” type by the Architect, the model is loaded into the Eclipse environment and all the subsequent steps are performed in it.

For the selection of such an environment we did not evaluate several existing tools as we did to select the tools for BP modeling and execution, since we have already been using the Eclipse environment and found that it provides the main characteristics required: free license, extension of functionalities in the form of plug-in development and several existing plug-ins providing other required functionality that can easily be integrated.

In particular, we have integrated the following plug-ins in our own distribution called Eclipse MINERVA design<sup>12</sup> built on top of the Eclipse Modeling distribution which includes EMF<sup>13</sup> for creating, managing and validating metamodels in .ecore format (which is similar to EMOF):

<sup>12</sup><http://alarcos.esi.uclm.es/MINERVA/TOOLS/>

<sup>13</sup><http://eclipse.org/modeling/emf/>

- **SAP<sup>14</sup> BPMN2 editor:** an implementation of the BPMN2 standard as an .ecore meta-model and a corresponding editor, which allows edition and validation of BPMN2 models against the metamodel
- **Eclipse MediniQVT<sup>15</sup> plug-in:** provides an editor and engine to define and execute QVT transformations in accordance with the QVT-Relations language
- **Eclipse UML Papyrus<sup>16</sup> plug-in:** provides a graphical environment for UML2 modeling, which allows the construction of UML profiles on top of it

In addition to integrating existing plug-ins we have developed two plug-ins to support functionalities that there were not provided by Eclipse:

- **Eclipse SoaML<sup>17</sup> plug-in:** an implementation of the SoaML standard on top of the Eclipse UML Papyrus plug-in which allows the modeling of services using SoaML, and the import and export of models in XMI format, along with UML2 modeling
- **Eclipse iS4BP<sup>18</sup> plug-in:** (insert Services for BP execution) provides a way to automatically populate a BPMN2 model with the corresponding invocations to the services generated from the SoaML model, to provide the basis for making it executable in a process engine

For the implementation of the defined services in the selected technology and to make the BPMN2 model executable in a BPMN2 process engine, another Eclipse distribution is needed which we called Eclipse for development, which will preferably be an Eclipse JEE distribution, or the Eclipse ModelPro for code generation. In Figure 9.3 the complete view of tools support provided by the Eclipse MINERVA design distribution is shown, along with the defined inputs and outputs.

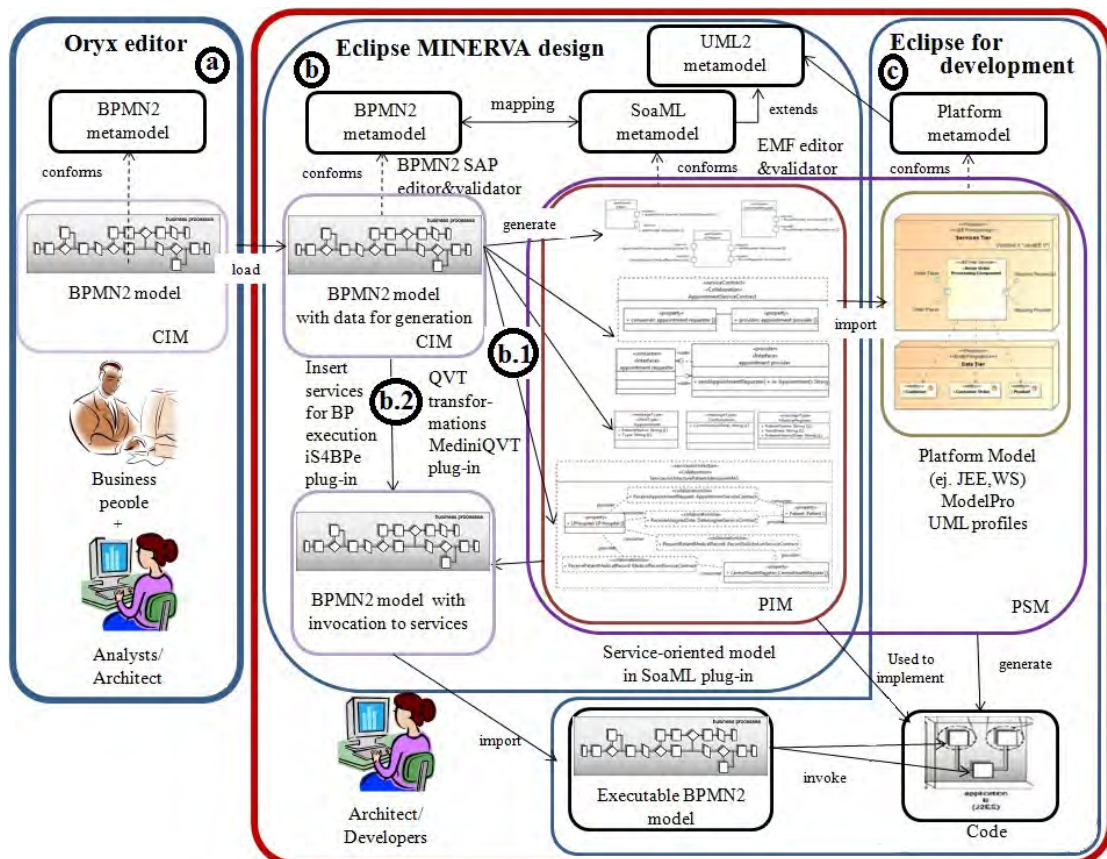


Figure 9.3.: Tools support for BP and services modeling and implementation

<sup>14</sup><http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/c04f0691-0a76-2d10-1098-ec518f7bdf68>

<sup>15</sup><http://projects.ikv.de/qvt/>

<sup>16</sup><http://www.eclipse.org/modeling/mdt/papyrus/>

<sup>17</sup><http://alarcos.esi.uclm.es/MINERVA/TOOLS/soamIPlugin.htm>

<sup>18</sup><http://alarcos.esi.uclm.es/MINERVA/TOOLS/iS4BP.htm>

The box (a) on the left of Figure 9.3 shows the modeling in BPMN2 in the Oryx editor as part of the activities in the Design&Analysis phase. It constitutes the main input for the Configuration phase which is supported by two Eclipse environments: (b) Eclipse MINERVA design distribution -which we provide as part of MINERVA- in which the BP model in BPMN2 format is loaded for the automatic generation of services, and (c) Eclipse for development -which is not provided as part of MINERVA- in which the services and the BPMN2 model are implemented.

In Eclipse MINERVA design shown in the box (b), the loaded BPMN2 model can be edited and validated by means of the SAP BPMN2 editor integrated. (b.1) shows the navigation from the loaded BPMN2 model on the right to generate the SoaML service models, by means of QVT transformations defined and executed in MediniQVT plug-in. To do this the BPMN2 model has to be transformed previously into XMI format which is not shown in this figure, as described in chapter 8. The SoaML model generated in XMI format is imported and visualized in the SoaML plug-in, which is part of our contribution. In addition, the SoaML and UML models can be edited and validated by means of EMF.

(b.2) shows the navigation from the loaded BPMN2 model to the bottom, extending it with information taken from the generated services in the SoaML model, by means of the iS4BPe plug-in, which is also part of our contribution. The iS4BPe inserts into the BPMN2 file the information for the invocation of services into the right activities, along with the complete definition of elements defined in the BPMN2 standard to be used for this invocation. It also inserts the information when the input is an XPD or a BPEL file, which have been obtained previously from the BPMN2 model. The output model can be imported into a suitable tool to add the rest of the information needed, for each specific process engine selected.

Eclipse for development corresponds to another (any) distribution of Eclipse required to implement services and the BPMN2 model for execution, which can be selected according to the context and needs of the organization. We do not provide such a distribution as this corresponds to the way the software team prefers to implement services based on the SoaML service model we generate with Eclipse MINERVA design. We separate the design and implementation environments to allow each team to focus on its own activities and artifacts which require different (and sometimes conflicting) Eclipse plug-ins to be integrated in the environment. Nevertheless, for Eclipse for development we suggest two main options:

- the first option is to generate code from the SoaML model, the ModelPro<sup>19</sup> Gold distribution can be used, which is an Eclipse JEE distribution with the MDA engine ModelPro providing generation for JEE and Web Services from SoaML service models. This option requires the integration of the Magic Draw tool which is a commercial tool, in which the SoaML model can be imported and extended for generation with information about the platform, by applying the JEE profile provided by ModelPro. A drawback of this option is that the version of the Eclipse environment provided is rather old (3.4 being the current version 3.7) so other plug-ins (such as the SoaML plug-in) are difficult to integrate, and the tool has not been updated since its last release.
- the second option is to implement services manually by means of an Eclipse JEE distribution, which provides several tools and generators for JEE and Web Services, which can be implemented based on the definitions in the SoaML model. The SoaML plug-in can also be installed in this environment, as long as the required elements are compatible (for example versions of EMF, GMF and Papyrus), and also other plug-ins such as the Activiti Designer for implementing the BPMN2 model for the Activiti engine. This plug-in is under development, so it does not yet provide graphically all the elements defined in the BPMN2 standard, but it provides support for the edition of the model in XML format and adds manually the information, from which the deployment artifacts are generated to be executed in the process engine.

---

<sup>19</sup><http://www.modeldriven.org/>

As a key tool from our contribution, in Figure 9.4 a screenshot of Eclipse MINERVA design is presented, in which the structure of projects given is shown on the left marked with: (1) a folder MINERVAdefs, in which three subfolders organized the Metamodels, QVTtransformations and qvtTraces for defining and executing transformations, and a folder MINERVAExample which in turn contains a models folder in which the example input BPMN2 model is provided, along with the XMI source and target models for the QVT transformations, and an SoaML folder which contains the result of importing the SoaML XMI model into the SoaML plug-in. In (2) the SoaML metamodel in .ecore format is shown, which uses the .ecore UML metamodel as can be seen at the bottom of the figure. In the following the plug-ins corresponding to our contribution and integrated into Eclipse MINERVA design are described.

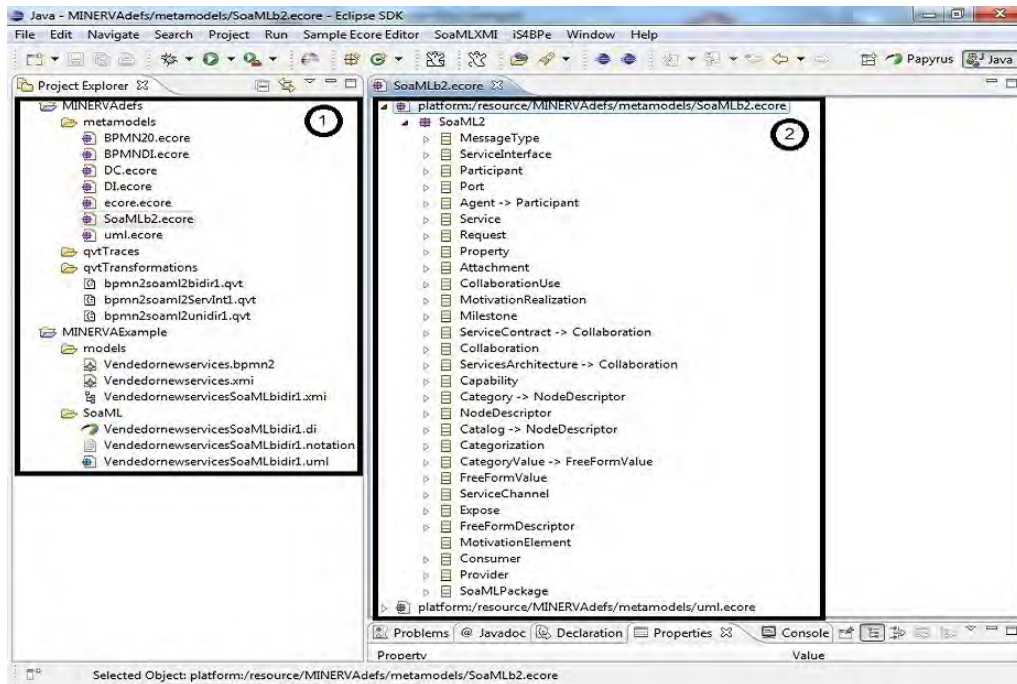


Figure 9.4.: Eclipse MINERVA design distribution screenshot

### 9.2.3.1. Eclipse SoaML plug-in

The Eclipse SoaML plug-in has been developed specifically to be integrated into the Eclipse MINERVA design distribution, to be able to import the XMI file which is the output of the QVT transformations, and visualize the generated service models. There are few implementations of SoaML<sup>20</sup> and they are mostly commercial, and only one can be used in the Eclipse environment but by integration with the tool<sup>21</sup> not as a plug-in.

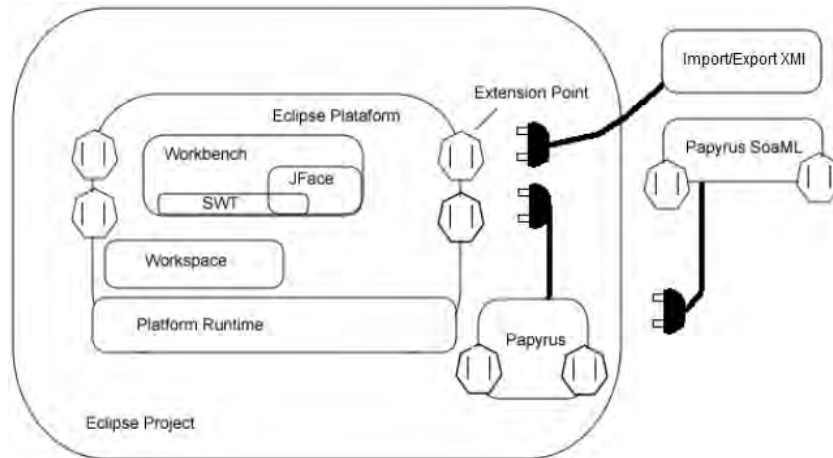
Several options were evaluated for the implementation of the plug-in, including the selected Papyrus, and others such as GMF<sup>22</sup> and UML2Tools<sup>23</sup>, choosing Papyrus as the base implementation of UML2 which also provides extension points to add specific UML2 profiles, as SoaML is. Each diagram in SoaML is built upon existing Papyrus UML elements, reusing and extending its capabilities with SoaML diagrams and stereotypes. The Import/Export XMI functionality is provided apart from Papyrus, as an Eclipse plug-in making it available from the general menu. In Figure 9.5 the Eclipse general Architecture with Papyrus and SoaML plug-ins is shown.

<sup>20</sup><http://www.omgwiki.org/SoaML/doku.php>

<sup>21</sup>[https://www.magicdraw.com/cameo\\_so](https://www.magicdraw.com/cameo_so)

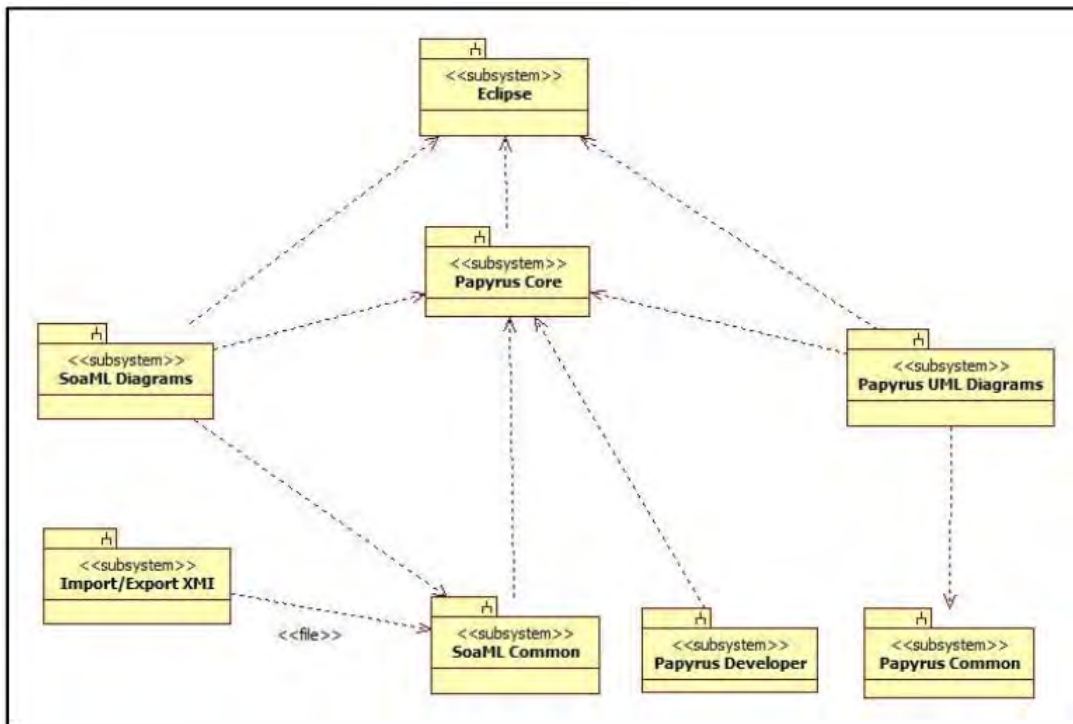
<sup>22</sup><http://www.eclipse.org/modeling/gmp/>

<sup>23</sup><http://wiki.eclipse.org/MDT-UML2Tools>



**Figure 9.5.:** Eclipse general Architecture with Papyrus and SoaML plug-ins

In Figure 9.6 the extension of Papyrus defined for the SoaML plug-in is shown, by means of the SoaML subsystems defined, existing Papyrus subsystems used and Eclipse.



**Figure 9.6.:** Subsystems of the SoaML Extended Papyrus Architecture

The subsystems Eclipse, Papyrus Core, Papyrus UML Diagrams and Papyrus Common are part of the Papyrus and Eclipse Architecture, which have been extended by adding the subsystems: Papyrus Developer, SoaML Common and SoaML Diagrams, and the Import/Export XMI subsystem. The Papyrus Developer subsystem accesses the Papyrus Core in order to add the application of the SoaML profile by default, and the new SoaML category of diagrams as an option when creating a new Papyrus model. As Papyrus provides each different type of diagrams in a different module (.jar) the same approach was followed for the SoaML plug-in, which is shown in Figure 9.7 in the components view for the solution.

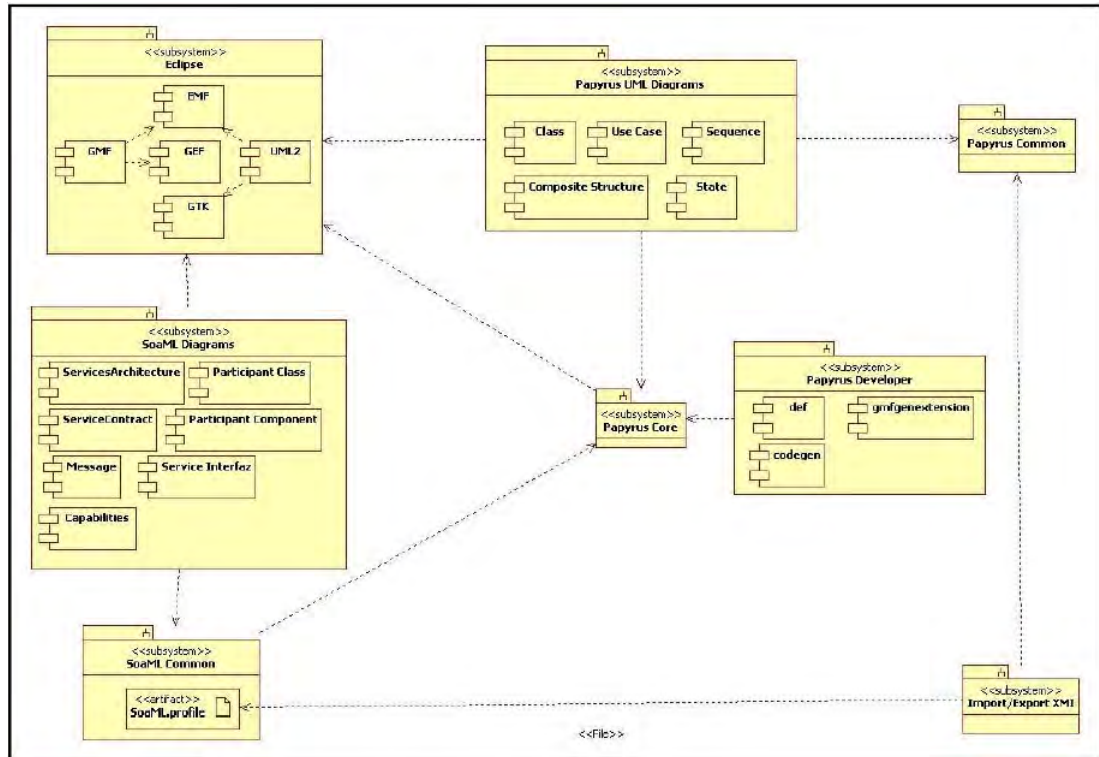


Figure 9.7.: Components view of the SoAML solution

Seven types of diagrams are provided in the Eclipse SoAML plug-in including: Services Architecture, Participants (class and component), Interfaces (class), Service Contract, Messages Types and Capabilities. Each diagram provides its own palette with the elements that can be used for the specific modeling, for example for the Services Architecture the collaboration is provided, along with the participant parts for the participants and the collaboration use for the contract use, as well as the role binding connecting them. Diagrams for the ServicesArchitecture, ServiceContract and Participants (Component) are based on the Papyrus Composite structure diagram editor, and diagrams for Participants (Class), Services Interfaces, Messages and Capabilities are based on the Papyrus Class diagram editor. The subsystems SoAML Common and SoAML diagrams -which includes the plug-ins corresponding to each diagram- are added to Papyrus as Eclipse plug-ins.

SoAML models constructed or imported in the Eclipse SoAML plug-in, are displayed using the Papyrus model explorer view as a tree that shows the elements corresponding to the visual model under construction. It allows us to assign the corresponding types for each element and checks restrictions on the elements and types used. The icons provided are based on the UML ones along with text explanations that help in the use of the elements. The XMI Import/Export facilities are defined in a specific Eclipse menu that provides intuitive use of the functions, and interoperability with other tools has been tested with Modelio <sup>24</sup> free edition v.1.2.1 which is no longer available. In Figure 9.8 a screenshot of the Eclipse MINERVA design distribution is presented, with a generated SoAML service model and showing the SoAML XMI Import/Export menu.

(1) the left part of Figure 9.8 shows the structure of the project PatientMAS in which two folders are defined: a models folder contains the input BPMN2 model and the source and target XMI models, and an soaml folder contains the SoAML model as imported by the SoAML plug-in from the target XMI model. In (2) the SoAML XMI model output of the transformation is shown in tree form in the Eclipse Ecore Reflective editor, with the structure we have defined for the model generated as presented in chapter 8. Navigating from it to the top (3) shows the Eclipse menu where the SoAML Import/Export options are displayed.

<sup>24</sup><http://www.modeliosoft.com/>

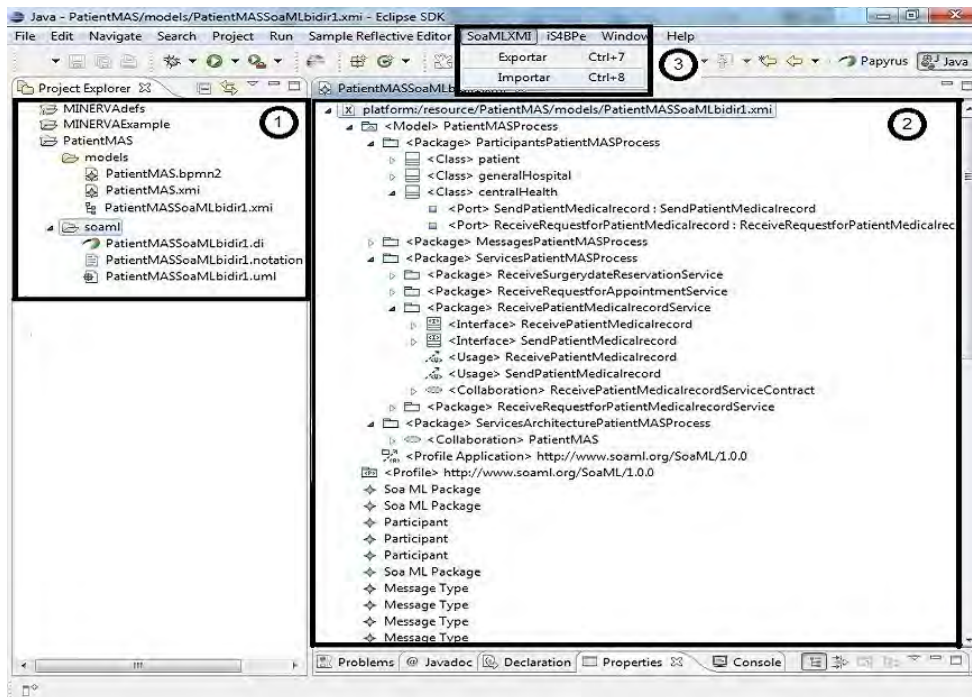


Figure 9.8.: Eclipse SoaML plug-in Import/Export and visualization with EMF

Once the SoaML XMI file has been imported in the plug-in, changing to the Papyrus perspective it is possible to visualize the Model Explorer of Papyrus on the left, and to construct the diagrams to show the elements generated based on the Papyrus model (.di). In Figure 9.9 the SoaML plug-in diagrams options are shown in the Papyrus diagram options, to select the Services Architecture Diagram within the selected corresponding package in the tree of the Model Explorer on the left.

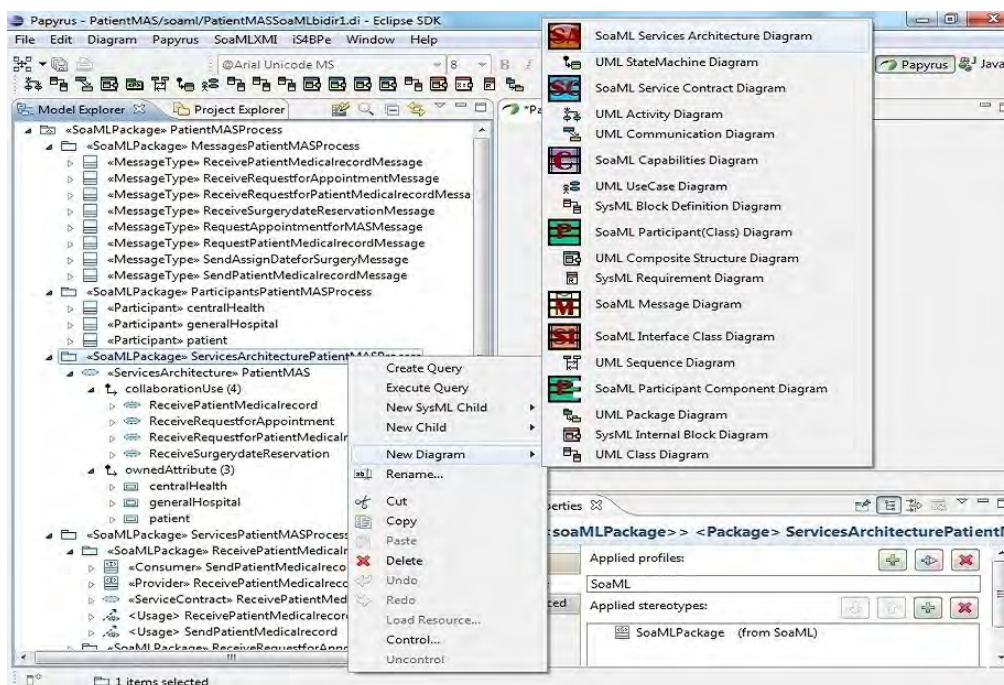


Figure 9.9.: Eclipse SoaML plug-in new ServicesArchitecture diagram layout

It can be seen that in the Papyrus Model Explorer the SoaML stereotypes are shown as part of the element rather than at the end of the model as EMF shows according to the organization of



the XMI file. In the options for new diagrams the seven types of diagrams provided are shown, and in the properties view the application of the SoaML profile can be seen.

After creating the Services Architecture diagram, it can be populated by dragging&dropping the elements generated, for example the ones corresponding to participants and services contracts and the role binding which automatically connects the associated elements. In Figure 9.10 part of the population of the Services Architecture diagram is shown as an example, showing participants and some generated services, along with the roles played for each participant within each service.

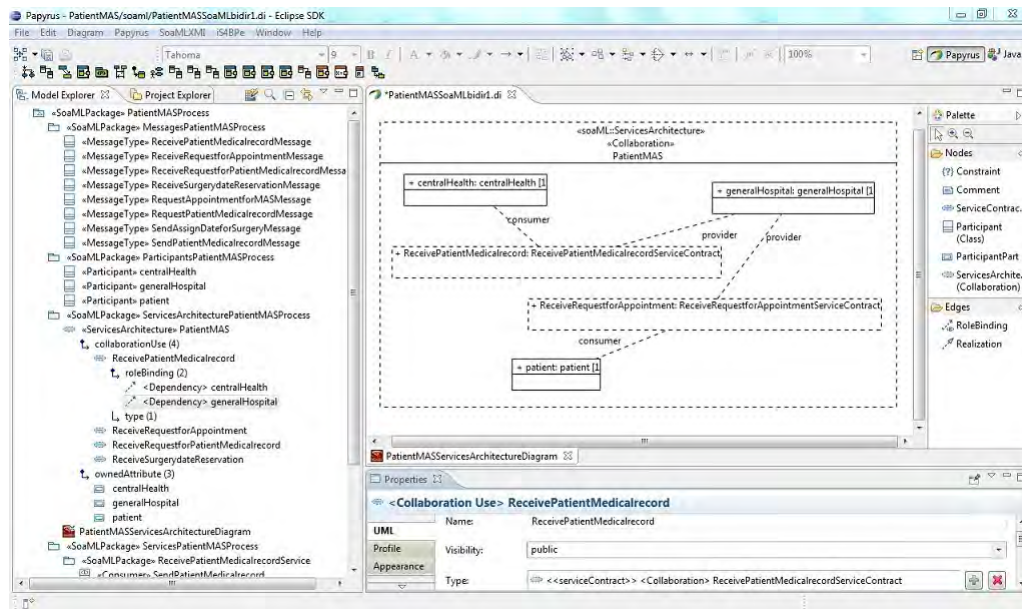


Figure 9.10.: Eclipse SoaML plug-in population of the ServicesArchitecture diagram

A demo in the form of a web video showing the main functionalities provided by the Eclipse SoaML plug-in can be seen in<sup>25</sup>.

### 9.2.3.2. Eclipse iS4BPpe plug-in

The iS4BPpe (insert Services for BP execution) plug-in has been developed specifically to be integrated into the Eclipse MINERVA design distribution, to insert into the file corresponding to the BPMN2 model (XPDL, BPEL or BPMN2) the services invocation as defined in the generated SoaML model, to help make it executable. iS4BPpe plug-in is a parser which uses jDOM<sup>26</sup> to locate the activities in the input file that corresponds to each generated service in the SoaML file, adding the information generated on operations, parameters and messages types according to the tags defined by each language (XPDL, BPEL or BPMN2).

For each language we provide a generic insert which consists in taking the corresponding file as input and only adding information which is required by the associated standard, and nothing more. The output file can be then imported in a designer for each language which in general is provided by the associated process engine, to add the rest of the information as required by the engine. We also planned to provide for each language a specific insertion of definitions and elements for each of the selected process engines, as presented in the next section, but as the time taken in the first one was more than expected, we decided to provide it for the BPMN2 engine, which is Activiti, and to leave the rest for future work. In Figure 9.11 a screenshot of Eclipse MINERVA design showing the iS4BPpe is presented.

<sup>25</sup><http://alarcos.esi.uclm.es/MINERVA/TOOLS/demoSoaMLplugin/DemoEclipseSoaMLplugin.htm>

<sup>26</sup><http://www.jdom.org/>

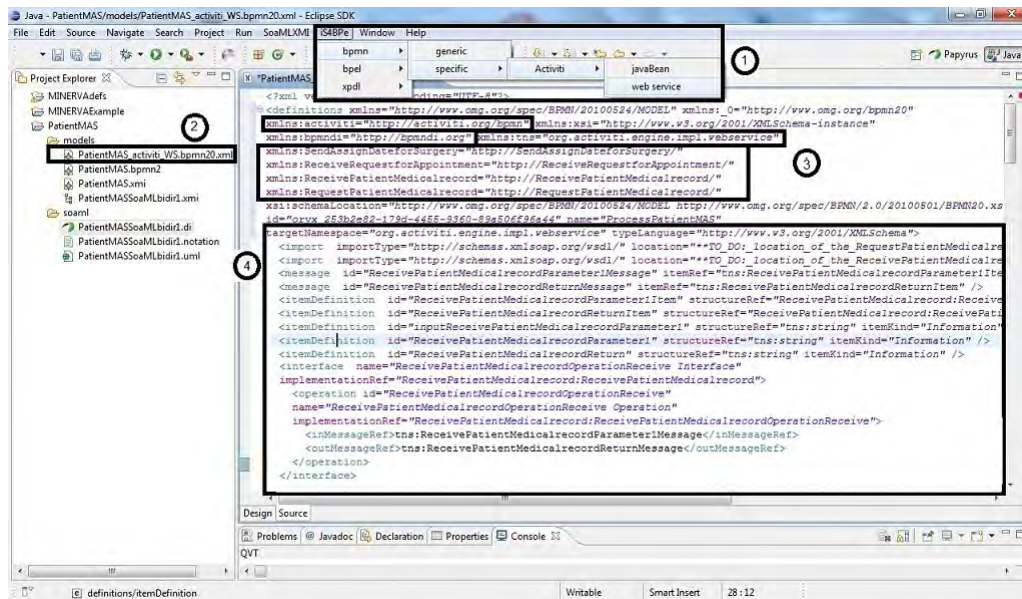


Figure 9.11.: Web Services Invocation generated for the Activiti engine

At the top of Figure 9.11 marked with (1), the menu of the iS4BP plug-in is shown presenting the options for the three languages: BPMN2, XPDL and BPEL. The first one is displayed showing the options for generic and specific generation for the Activiti process engine, and in it two of the options it provides for invoking services: Web Services as defined in the BPMN2 standard, and via a java class that has to be further implemented to invoke the service desired. As can be noticed, instantiating the insertion for a specific engine provides different options which are related for the specific implementation the engine provides. On the left marked with (2), the new file generated in BPMN2 format with the invocation to services inserted is shown, as we preserve the input BPMN2 file so it can still be used in the environment as is. Moving to the right marked with (3), the header of the XML file is shown, including the specific xmlns definitions to recognize the file as required by Activiti, and defining the namespaces for each inserted invocation. Moving down to the bottom marked with (4), part of the declarations for the invocation of Web Services as defined by the BPMN2 standard are shown, defining the import for the WSDL for each invocation (with a “TODO” insertion to be changed with the actual address of the service), the definition of messages, Items and Interfaces with operations and references to messages in and out, to be used in the invocation inserted in the Service Task, which is shown in Figure 9.12.

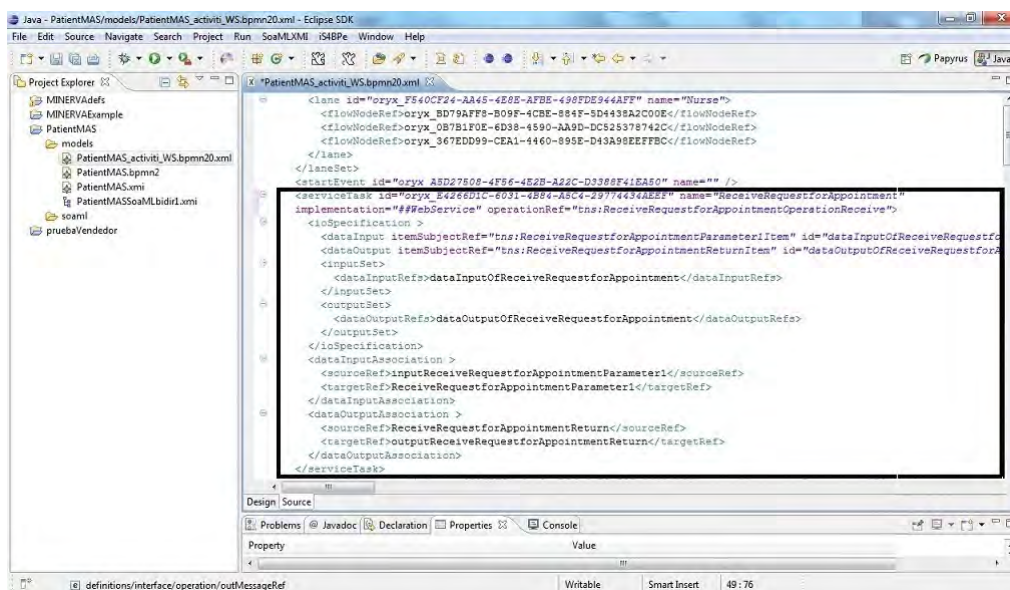


Figure 9.12.: ServiceTasks inserted information for the Activiti engine

### 9.2.4. Enactment phase

Business Process execution is another key activity if we want the organization to be process driven. The BP models specified by business and software people have to be extended with information for execution, in order to be interpreted by a process engine executing the corresponding language. At the start of this thesis the standards favored most for BP execution were XPDL and BPEL [Hornung et al., 2006], as BPMN was not executable at that time, so a transformation was needed to navigate from a BPMN model to an XPDL or WS-BPEL model to be able to execute it, and a process engine for executing each language had to be selected.

The evaluation was carried out by two student<sup>27</sup> groups, one evaluating XPDL process engines, and the other evaluating WS-BPEL, based on a thorough examination of the characteristics of the XPDL implementation tools listed in WfMC<sup>28</sup> and the WS-BPEL implementation tools listed in OASIS<sup>29</sup>, among other sources. The procedure defined for the evaluation of characteristics was defined as follows:

1. define groups A and B of characteristics to be evaluated, as presented in section 9.1, taking into account several sources (ISO SQUARE [ISO, 2005-2011], HGCR document, among others) and existing evaluations (workflow patterns evaluations<sup>30</sup>) and define the weight of each characteristic for group B (in group A all characteristics are mandatory).
2. take as input the WfMC XPDL implementation list and OASIS WS-BPEL implementation list, among other reference lists.
3. evaluate group A of characteristics on each engine by reading the documentation provided by each one, to select only the appropriate process engine (i.e. XPDL as language, WS-BPEL as language, license type, stable version, among others).
4. if the result of the assessment based on group A exceeds six tools, the characteristics provided by the engines were compared with each other and the ones ranked most were chosen.
5. evaluate the group B of characteristics from the resulting list of engines to obtain a ranked list of the characteristics provided by each process engine.

When the evaluation was coming to an end, the BPMN2 standard was released and several pioneer tools released their implementations of BPMN2 process engines, so we decided to include BPMN2 execution as well. An example of the characteristics evaluated as part of group B is presented in Table 9.3, which are some of the most interesting ones, showing only the characteristics defined but not the grouping of them as this was defined differently by the two groups. The XPDL, WS-BPEL and BPMN2 columns show to which of the engines the characteristic applied.

**Table 9.3.:** Selection of characteristics evaluated for each type of engine

Characteristic	Description	Evaluated for		
		XPDL	WS-BPEL	BPMN2
1	Control flow routing	X	X	X
2	Business rules	X	X	X
3	Activity assignment by roles	X	–	X
4	List of activities	X	–	X
5	Definitions of types of data and objects	X	–	X
6	Functionality adding by imbibed code	X	–	X
7	Platform independence	X	–	X
8	Logging and audit information facilities	X	X	X
9	Import/Export XPDL	X	–	X
10	Import/Export WS-BPEL	–	X	X
11	Import/Export BPMN2	–	–	X

<sup>27</sup>final project in a five year Computer Science degree

<sup>28</sup><http://www.wfmc.org/>

<sup>29</sup><http://www.oasis-open.org/>

<sup>30</sup><http://www.workflowpatterns.com/evaluations/>

Characteristic	Description	Evaluated for		
		XPDL	WS-BPEL	BPMN2
12	Integration with other tools (e-mails, DBs)	X	X	X
13	Multiple version execution	X	X	X
14	Exporting reports	X	X	X
15	Users management	X	X	X
16	Roles management	X	X	X
17	Profiles management	X	X	X
18	Fault recovery	X	X	X
19	Rollback process execution	–	X	–
20	Number of execution BPs	X	X	X
21	Number of executing instances of each BP	–	X	–
22	Number of concurrent users	X	X	X

Finally the following process engines to be evaluated were selected for XPDL, WS-BPEL and BPMN2, from which one for each language would be selected for integration into MINERVA:

- **XPDL process engines:** Bonita<sup>31</sup>, Enhydra Shark<sup>32</sup> (now part of Together), Joget<sup>33</sup>, OBE<sup>34</sup>, WfmOpen<sup>35</sup>
- **WS-BPEL process engines:** Orchestra<sup>36</sup>, Intalio<sup>37</sup> Community edition, Apache ODE<sup>38</sup>, Riftsaw<sup>39</sup>, jBPM<sup>40</sup>
- **BPMN2 process engines:** Activiti<sup>41</sup>, jBPM5<sup>42</sup>.

As a result of the evaluation, one engine for each language was selected for integration into MINERVA: Bonita for XPDL, Intalio community edition for WS-BPEL and Activiti for BPMN2:

- Bonita provides most of the characteristics defined for a process engine, including several tools that allow the modeling, implementation, execution and evaluation of BPs to be supported, also providing import/export of models in XPDL format. It allows the execution of BPs with complex logic and provides good integration with several other tools, its environment both for implementation and execution is user friendly, and the execution is web. It also allows, among other things, the definition of roles and profiles for participants, working list management, alerts for activity deadlines, monitoring screens with execution information, and registering of execution data supporting several data bases. It was initiated in 2001 and has been continuously improved and updated since then.
- Intalio provides several of the characteristics defined for a process engine, including import of models in WS-BPEL format (when it was evaluated it did not present WS-BPEL export facilities). Its implementation and execution environment are friendly, the execution is web, and it allows the management of BP cases execution by means of several possible actions such as suspend, resume or terminate. It also provides definition of roles for participants, registering of execution data supporting several data bases, although no information on the user executing activities is registered, which is a main drawback of the engine. It was launched in 1999 and has been continuously improved and updated.

<sup>31</sup><http://www.bonitasoft.com/>

<sup>32</sup><http://www.together.at/prod/workflow>

<sup>33</sup><http://www.joget.org/>

<sup>34</sup><http://obe.sourceforge.net/>

<sup>35</sup><http://wfmopen.sourceforge.net/>

<sup>36</sup><http://orchestra.ow2.org>

<sup>37</sup><http://community.intalio.com/>

<sup>38</sup><http://ode.apache.org/>

<sup>39</sup><http://www.jboss.org/riftsaw>

<sup>40</sup><http://www.jboss.org/jbpm>

<sup>41</sup><http://www.activiti.org/>

<sup>42</sup><http://www.jboss.org/jbpm>

- Activiti provides BPMN2 execution with high adherence to the BPMN2 standard, as well as some “native” options with activiti extensions. The implementation and execution environments are friendly and the execution environment is web, allowing, among other things, the definition of roles to be assigned to participants, working list management, monitoring of the execution progress of the BP cases, registering of execution data supporting several data bases. It was first released in the first half of 2010 in its 5.0 version as their developers came from jBPM versions 1 to 4 (launched at the beginning of this millennium) and continued the implementation with Activiti. Although we selected Activiti on the basis of our evaluation of its characteristics, jBPM5 also provides many similar ones now, and can also be integrated.

Interoperability tests were carried out to assess whether a BPMN2 model specified in a different modeler tool, such as Bizagi, could be exported to XPDL or BPEL and imported in Bonita and Intalio respectively, to be executed. These were also applied for BPMN2 models specified in a modeler to see if the BPMN2 file could be imported in Activiti.

For the first cases, transformations were carried out by means of the tools available, from BPMN2 to XPDL using the Bizagi process modeler, from BPMN2 to BPEL using BP incubator transformations<sup>43</sup>, but none of these importations were successful either in Bonita or Intalio. For the BPMN2 interoperability the BP modeled in Oryx was imported in the Activiti process engine, but as the engine only supports BPs defined in one pool, only the one corresponding to the organization view was kept, and the pools corresponding to external participants and the associated collaboration were deleted. The resulting BPMN2 model was validated using BP incubator, and in this way, the BPMN2 model was correctly imported and instantiated in Activiti.

Regarding BPMN2 interoperability it is expected that in the near future, if tools implement the standard, few changes would have to be made to be able to import BPMN2 files between different process engines to make them executable. Although we selected these process engines to support the enactment of BPs in a process engine, if there is one already in the organization it can be integrated with the previous tools we have defined for modeling and implementing BPs and services, following the guides tools provide for making the BPs executable and for invoking services from it.

Interoperability tests were also carried out to evaluate the capacities each process engine provides for registering data about the execution of BP instances, as we need to collect the data defined in the base measures to calculate the execution measures defined in BPEMM. In Table 9.4 the data that can be obtained from each process engine is shown.

**Table 9.4.:** BP engines facilities for registering execution data for measures calculation

Execution data registered	BP Engine		
	Activiti	Bonita	Intalio
BP id	Y	Y	Y
BP case id	Y	Y	Y
BP case end state	Y	Y	Y
Activity id	Y	Y	Y
Activity instance id	Y	Y	Y
Activity user	Y	Y	N
Activity ET	N	Y	Y
Activity ST	Y	Y	Y
Activity CT	Y	Y	Y

Each process engines registers data in a data base according to the tables schema each defines, so to extract the data several queries had to be performed to obtain the data desired. In general the .csv file option is provided to export the data, which can be transformed and rearranged to be used for further analysis. To obtain the MXML format representation, tools such as ProMImport<sup>44</sup> framework or Fluxicon Nitro<sup>45</sup> can be used, which allow several format inputs for execution data from different process engines and produce as output the corresponding MXML file, which is shown in the case study described in chapter 10.

<sup>43</sup><http://businessprocessincubator.com/>

<sup>44</sup><http://www.promtools.org/promimport/>

<sup>45</sup><http://fluxicon.com/nitro/>

### 9.2.5. Evaluation phase

Business Process execution analysis is another key activity if we want to be able to find improvement opportunities for the BPs based on the measurement results from their execution. A tool that can be used and understood by the business and software areas is needed, to support the analysis of the registered data, in order to be able to evaluate the execution of BPs from different perspectives and with different objectives.

For the selection of such a tool we did not evaluate several existing tools as we did to select the tools for BP modeling and execution, since we have already been using the ProM<sup>46</sup> framework and have found that its new release ProM6<sup>47</sup> provides the main characteristics required: free license, extension of functionalities in the form of plug-in development and several existing plug-ins providing the implementation of many process mining techniques. In addition to integrating existing plug-ins we have developed a plug-in to support the definitions of execution measures in BPEMM:

- **ProM BPEMM plug-in:** an implementation of the BPEMM execution measurement model to provide support for the analysis of BP execution based on the execution measures in the defined views, dimensions and hierarchy of BPEMM. As it is still a prototype at the moment, we can not release it yet, but we expect to do so by mid 2012.

As presented in chapter 4, the ProM framework is a generic open-source framework for implementing process mining tools in a standard environment, based on event logs from the BP execution specified in XES or MXML format. ProM was released in its first version by 2004 and currently provides more than 170 plug-ins, and at present it has a wide community of supporters. It also provides the import of and the conversion between several process modeling languages, such as: Petri nets, EPC, BPMN, among others.

#### 9.2.5.1. ProM BPEMM plug-in

The ProM BPEMM plug-in has been developed specifically for integration into the ProM framework, to provide support for the analysis of BPEMM execution measures, based on the views, dimensions and hierarchy defined. Although ProM provides several plug-ins these are mainly focused on the three process mining perspectives supported: discovering BP models from execution; checking conformance between an existing BP model and its real execution; and extending a BP model with execution information. For the analysis of other execution aspects such as execution times, among others, only two plug-ins are provided: the Basic Performance Analysis, and the Performance Analysis with Petri net, which present mostly performance information and the second needs a Petri net model of the BP as input.

Adding the ProM BPEMM plug-in will provide an integral view of the data from the execution of the BP, allowing business people to analyze it by taking into account several perspectives in an integrated environment. The ProM BPEMM plug-in needs three files as input:

- the execution event log in MXML format obtained from the execution data registered in the process engines, as described in the previous section
- a configuration file in XML, because as described in chapter 6 some context data can not be obtained from the BPs execution, such as the salary of each participant in the BP, or the definition of the “successful branch” for the execution of the BP, so we defined a configuration file to be produced to include this data.
- the BPMN2 file in XML format, because as described in chapter 3 the BP model is needed for the calculation of some measures, for example to know that some activities are performed in parallel.

---

<sup>46</sup><http://www.processmining.org/prom/start>

<sup>47</sup><http://www.promtools.org/prom6/>

In Figure 9.15 the ProM BPEMM plug-in definitions are presented, showing the three input files required with the output being the BPEMM measurement panel in which the execution measures are calculated and visualized for the BP cases included in the execution event log.

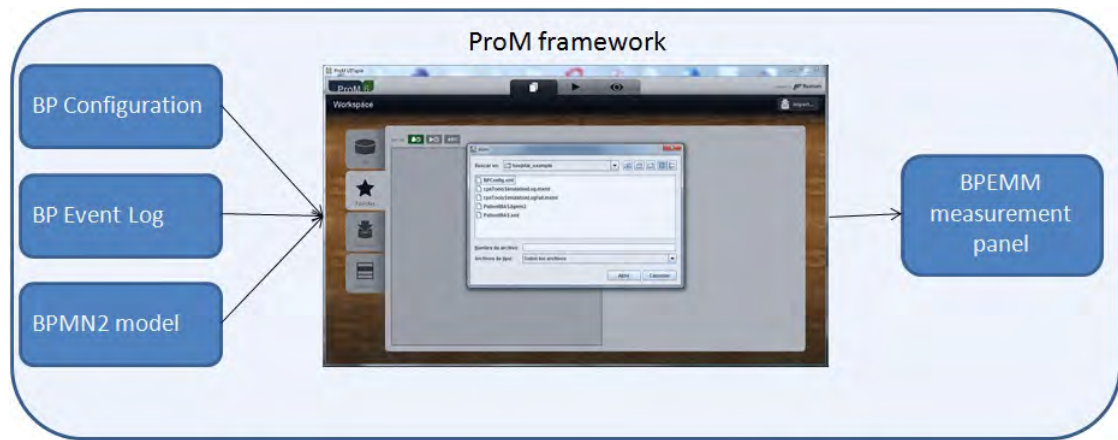


Figure 9.13.: ProM BPEMM plug-in definitions

The configuration file is defined by an XMLSchema which allows the XML files to be constructed and validated against its definition. The XMLSchema defined and an example of a configuration file used for the processing of BPEMM execution measures are shown in Figure 9.14. It can be seen that information about the execution measures is also provided, such as the decision criteria for defining the ranks for each indicator, which are defined with labels so they can be determined for each organization and each specific BP. Based on this information, when showing the measurement results for an indicator, the semaphore color can be assigned to show the meaning of the value.

<pre> 1 &lt;?xml version="1.0" encoding="UTF-8"?&gt; 2 &lt;schema xmlns="http://www.w3.org/2001/XMLSchema" 3   targetNamespace="http://www.example.org/BPConfig" 4   xmlns:tns="http://www.example.org/BPConfig"&gt; 5   &lt;element name="BPConfig" type="tns:BPConfigType"/&gt;&lt;/element&gt; 6   &lt;complexType name="DecisionCriteria"&gt; 7     &lt;attribute name="name" type="string"/&gt;&lt;/attribute&gt; 8     &lt;attribute name="minlevel" type="double"/&gt;&lt;/attribute&gt; 9     &lt;attribute name="maxlevel" type="double"/&gt;&lt;/attribute&gt; 10  &lt;/complexType&gt; 11  &lt;complexType name="ResourcePerRole"&gt; 12    &lt;attribute name="role" type="string"/&gt;&lt;/attribute&gt; 13    &lt;attribute name="person" type="string"/&gt;&lt;/attribute&gt; 14  &lt;/complexType&gt; 15  &lt;complexType name="CostPerResource"&gt; 16    &lt;attribute name="person" type="string"/&gt;&lt;/attribute&gt; 17    &lt;attribute name="cost" type="double"/&gt;&lt;/attribute&gt; 18  &lt;/complexType&gt; 19  &lt;complexType name="BranchSuccessful"&gt; 20    &lt;attribute name="activity" type="string"/&gt;&lt;/attribute&gt; 21  &lt;/complexType&gt; 22  &lt;complexType name="BranchUnsuccessful"&gt; 23    &lt;attribute name="activity" type="string"/&gt;&lt;/attribute&gt; 24  &lt;/complexType&gt; 25  &lt;complexType name="BPConfigType"&gt; 26    &lt;sequence&gt; 27      &lt;element name="DecisionCriteria" type="tns:DecisionCriteria" 28        maxOccurs="unbounded" minOccurs="1"/&gt; 29    &lt;/element&gt; 30    &lt;element name="ResourcePerRole" type="tns:ResourcePerRole" 31      maxOccurs="1" minOccurs="0"/&gt; 32    &lt;/element&gt; 33    &lt;element name="CostPerResource" type="tns:CostPerResource" 34      maxOccurs="unbounded" minOccurs="0"/&gt; 35    &lt;/element&gt; 36    &lt;element name="BranchSuccessful" type="tns:BranchSuccessful" 37      maxOccurs="unbounded" minOccurs="0"/&gt; 38    &lt;/element&gt; 39    &lt;element name="BranchUnsuccessful" 40      type="tns:BranchUnsuccessful" maxOccurs="unbounded" minOccurs="0"/&gt; 41    &lt;/element&gt; 42  &lt;/sequence&gt; 43  &lt;/complexType&gt; 44 &lt;/schema&gt; </pre>	<pre> 1 &lt;?xml version="1.0" encoding="UTF-8"?&gt; 2 &lt;tns:BPConfig xmlns:tns="http://www.example.org/BPConfig" 3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 4   xsi:schemaLocation="http://www.example.org/BPConfig BPConfig.xsd"&gt; 5   &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="TWaT"/&gt; 6   &lt;DecisionCriteria maxlevel="60.0" minlevel="40.0" name="TWoT"/&gt; 7   &lt;DecisionCriteria maxlevel="100.0" minlevel="70.0" name="BFTI"/&gt; 8   &lt;DecisionCriteria maxlevel="1.0" minlevel="0.5" name="TTI"/&gt; 9   &lt;DecisionCriteria maxlevel="60.0" minlevel="40.0" name="FWoT"/&gt; 10  &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="FWaT"/&gt; 11  &lt;DecisionCriteria maxlevel="90.0" minlevel="80.0" name="ABFTI"/&gt; 12  &lt;DecisionCriteria maxlevel="60.0" minlevel="40.0" name="ABFOTI"/&gt; 13  &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="ABFHaT"/&gt; 14  &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="AATHoT"/&gt; 15  &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="AATWaT"/&gt; 16  &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="AATI"/&gt; 17  &lt;DecisionCriteria maxlevel="1.0" minlevel="0.5" name="ATI"/&gt; 18  &lt;DecisionCriteria maxlevel="40.0" minlevel="20.0" name="FBFSB"/&gt; 19  &lt;DecisionCriteria maxlevel="1.0" minlevel="0.5" name="FBFUSB"/&gt; 20  &lt;ResourcePerRole person="Mary" role="Secretary"/&gt; 21  &lt;ResourcePerRole person="John" role="Secretary"/&gt; 22  &lt;ResourcePerRole person="System" role="Secretary"/&gt; 23  &lt;ResourcePerRole person="Service" role="Secretary"/&gt; 24  &lt;ResourcePerRole person="Lisa" role="Nurse"/&gt; 25  &lt;ResourcePerRole person="Sue" role="Nurse"/&gt; 26  &lt;ResourcePerRole person="System" role="Nurse"/&gt; 27  &lt;ResourcePerRole person="Service" role="Nurse"/&gt; 28  &lt;CostPerResource cost="12.4" person="Mary"/&gt; 29  &lt;CostPerResource cost="196.0" person="John"/&gt; 30  &lt;CostPerResource cost="134.0" person="Lisa"/&gt; 31  &lt;CostPerResource cost="123.4" person="Sue"/&gt; 32  &lt;BranchSuccessful activity="Register patient for MAS"/&gt; 33  &lt;BranchSuccessful activity="Give clothes to change for MAS"/&gt; 34  &lt;BranchSuccessful activity="Assign place for MAS"/&gt; 35  &lt;BranchSuccessful activity="Give information about MAS"/&gt; 36  &lt;BranchUnsuccessful activity="Inform patient about problem"/&gt; 37  &lt;BranchUnsuccessful activity="Cancel surgery"/&gt; 38 &lt;/tns:BPConfig&gt; </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figure 9.14.: XMLSchema (1) and example (2) of BP Configuration file ProM BPEMM plug-in

After importing the three files required into the ProM framework, two steps have to be performed: (1) select the event log file and apply the BPEMM plug-in to generate a process file from it, and (2) select the file obtained from the event log in the first step, the configuration file and the graph file obtained when importing the BPMN2 model file, and apply the BPMeasurement plug-in to generate the measurement panel. In Figure 9.15 a screenshot of the ProM plug-in is shown, where it can be seen that the measurement panel shows at the top the three views from BPEMM (Generic

BP, Lean BP, Services), then for each one the measures for each corresponding dimension, and finally the desired level of analysis: BP, BP cases or Activity instances.



Figure 9.15.: ProM BPEMM plug-in example for BP cases option

In Figure 9.15 the Generic View → Throughput time → all BP cases selection is shown for the execution of the adapted “Patient MAS” that is described in chapter 10. The Throughput Time (TT) for each BP case is presented graphically on the right, also showing the Total Working Time and Total Waiting Time. On the left, the results of the indicators are shown, along with a colored semaphore showing the associated meaning (Green = OK, Yellow = Warning, Red = Problems). A view on each BP case execution is obtained by changing the Measure level to each BP case, as shown in Figure 9.16.



Figure 9.16.: ProM BPEMM plug-in example for each BP case option

In Figure 9.16 the summary of the activities execution in the selected BP case is shown graphically



in the same way as for BP cases before. For each activity executed in the BP case, the Total execution time is shown along with the Activity Working Time and Activity Waiting Time. Another view for each activity is provided by taking into account all the BP cases in which it has been executed, as shown in Figure 9.17. This view allows the summarized behavior of the activity to be analyzed, as shown on the left with the defined indicators, and specifically in each BP case where it has been executed.



**Figure 9.17.:** ProM BPEMM plug-in example for an activity through all BP cases

Although the ProM BPEMM plug-in has been defined taking into account all the elements required to be able to calculate and present all the BPEMM execution measures, for the time being we have implemented a prototype that is a proof of concept of the plug-in only for the time dimension of the BP Generic view, specifically the Throughput Time (TT) as presented before. We have decided to start with this implementation as in the ProM community several event logs examples are available which allows the plug-in to be tested with various BPs executions. The implementation of the rest of the execution measures of BPEMM have been left for future work.

### 9.2.6. BPCIP lifecycle guidance tool support

As presented in chapter 5 and chapter 7 tool support is also provided for the guidance through the method of work throughout the BPCIP lifecycle, by means of the implementation of BPCIP and BPSOM in EPF Composer as method plug-ins. These plug-ins can be downloaded from the MINERVA site<sup>48</sup> as has already been mentioned in chapter 5 and chapter 7 respectively, to generate the corresponding Web Site for publication within the organization, to be easily accessed and used by all employees. This allows all the people involved to look for the description of activities and artifacts, along with responsible and participant roles, thus making it easier to carry out the work defined.

The method plug-ins also allow the integration with other processes defined in such a way, for example, it makes it possible to add BPSOM to the base software development process used in the organization, extending it with specific activities for developing services systems from BPs. As an example both of method plug-ins and Web Sites generated have been presented in the corresponding chapters, and the complete Web Sites are provided in Appendix B, we do not reproduce this here again. The purpose of mentioning them here is to state that they are also part of the tool support dimension defined in MINERVA to support the methodological definitions of the framework.

<sup>48</sup><http://alarcos.esi.uclm.es/MINERVA/>

### 9.3. Conclusions

In this Chapter the tool support defined for MINERVA has been presented, which provides the tools with the functionalities required to allow the defined method of work through the BPCIP lifecycle to be performed. The definition of tools was carried out based on the premise of reusing existing tools providing the required functionalities as much as possible, and only developing our own when no such tool exists.

For each BPCIP phase the evaluations of several existing tools have been presented along with the procedures followed in the assessments, to choose the tools that were the closest fit to MINERVA definitions. In some cases, when a tool providing the required functionalities was known it was chosen without evaluating others, justifying its integration. What is more, guidelines are provided for the integration of other existing tools if they are already used in the organization, such as for modeling or executing BPs. The functionality required to fit the rest of the tools defined in MINERVA is specified, for example in a BPMN2 modeling tool to be able to export in BPMN2 format as defined in the standard.

The result is a complete set of tools and guidance for working within the MINERVA framework, which supports the definitions made and provides an easy way of following and performing the activities defined by BPCIP and BPSOM, producing the associated artifacts.

## Chapter 10.

# MINERVA framework validation

This Chapter describes the validation of the MINERVA framework carried out to assess key elements of the proposal by means of an experiment for the QVT transformations defined and two case studies to assess the feasibility of applying BPCIP and BPSOM in organizations.

The Chapter is organized as follows: in section 10.1 the motivation for the three validations carried out is presented; these are presented in further detail in: section 10.2 which presents the experiment to assess the QVT transformations defined for the generation of SoaML service models from BPMN2 models; section 10.3 which describes a case study carried out to assess the applicability of BPSOM including the model driven approach, and in section 10.4 which describes a case study carried out to assess the improvement and execution measurement approach defined by BPCIP. Finally in section 10.5 conclusions for the chapter are discussed.

The contents of this chapter complement the contents in the following chapters: chapter 5 which presents the Business Process Continuous Improvement Process (BPCIP) defining the complete BP lifecycle from modeling to improving the BPs, chapter 6 which describes the Business Process Execution Measurement Model (BPEMM) defining BP and service execution measures to guide the measurement effort within the defined lifecycle, chapter 7 which presents the Business Process Service Oriented Methodology (BPSOM) to guide the development of service-oriented systems from BPs with a model-driven approach, chapter 8 which describes the MDA approach to generate SoaML service models from BPMN2 models, and chapter 9 which presents the tool support defined for MINERVA.

### 10.1. Introduction

MINERVA framework will be validated by means of a case study, but we also wanted to assess the definition of the QVT transformations with an experiment to gain more insight into the appropriateness and usability of the service models generated from the point of view of software engineers with experience in software modeling, so we also defined and carried out an experiment to validate the QVT transformations. We have assessed the Suitability (as sub-characteristic of Functionality as defined in ISO 9126 [ISO, 2001]) of the transformations, that is of the generated service models regarding the source BP model, with respect to what the user would design by him/herself, and the Understandability (as sub-characteristic of Usability as defined in ISO 9126 [ISO, 2001]) of the generated service models. The validation of the QVT transformations by means of the experiment is described in section 10.2.

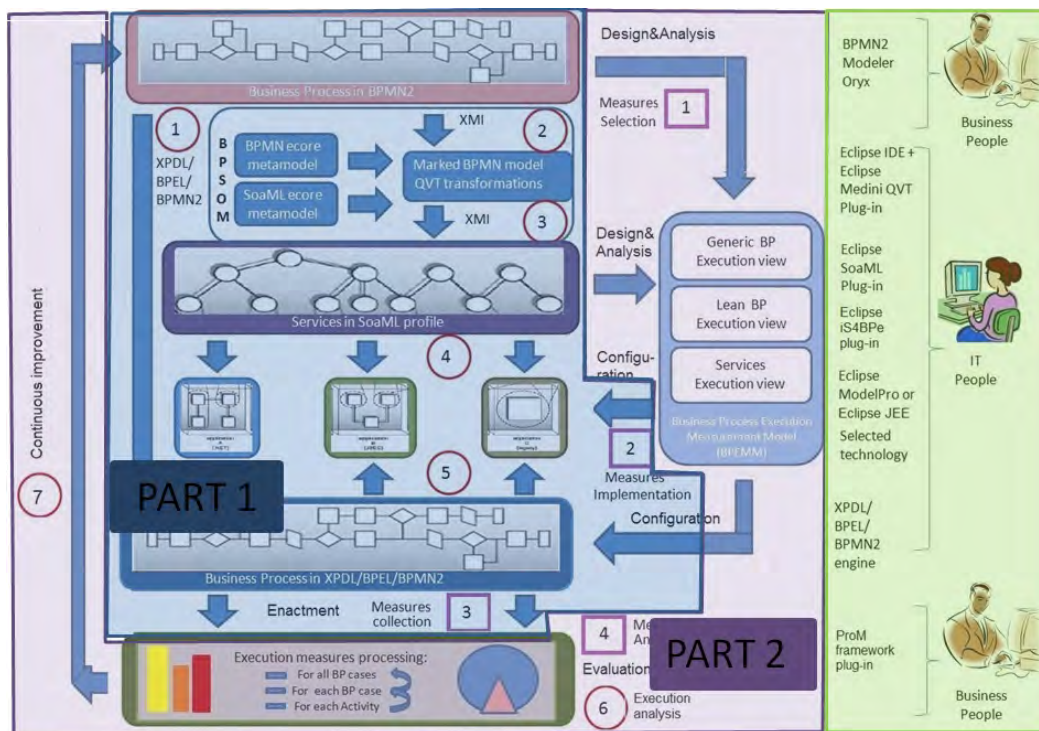
With regard to the validation of MINERVA framework, it was initially planned to be carried out as a case study in a project between the Alarcos Research Group and the Hospital General de Ciudad Real<sup>1</sup> (HGCR, General Hospital of Ciudad Real), Spain, in which several BP models had been specified previously in a joint work with the quality group and several users from the HGCR. In the context of this project and the work in MINERVA definition, we have defined the evaluation characteristics for BPMS selection which were also applied to select the BP modelers and BP engines for MINERVA, as described in chapter 9. The case study was defined to be carried out in two parts:

---

<sup>1</sup><http://www.hgucr.es/>

1. **Part 1:** the IT area of the HGCR should follow the BPSOM and automatic generation guidelines to develop a prototype of a selected BP model in the Hospital, from the ones that had previously been modeled. This part should generate as output the BP model in executable form to be deployed in the selected process engine, as well as the corresponding services to be implemented and then invoked from the executing BP, registering the defined measures, covering the steps 1 to 5 in the red circles and steps 2 and 3 in the purple squares the MINERVA method of work.
2. **Part 2:** the execution measurement approach of BPEMM and the measurement and improvement activities defined by BPCIP will be carried out by the HGCR quality group and M.D. experts in the BP, in order to define the execution measures for the selected BP and to asses the execution of the developed prototype in part 1, to find improvement opportunities and to generate a new version to be deployed and evaluated, following the defined improvement activities, covering the steps 6 and 7 in the red circles and steps 1 and 4 in the purple squares.

Figure 10.1 shows the coverage of the parts defined for the case study to validate the complete MINERVA method of work through BPCIP as presented in chapter 4, to give the reader the context of the definitions presented. The tools support and the defined roles are marked in another color just to indicate that they are included in the realization of each of the defined validations. Steps 1 to 5 in the red circles and steps 2 and 3 in the purple squares will be covered by the first part of the case study and the steps 6 and 7 in the red circles and steps 1 and 4 in the purple squares will be covered by the second part.



**Figure 10.1.:** MINERVA method of work coverage by the case study defined initially

As the infrastructure and the IT area of the HGCR was not available to carry out the first part of the case study as defined, within the timespan of this thesis, we decided to overcome this by splitting the first part of the case study in two sub-parts, based on the possibilities we had at that particular moment in time, and re-defined the part 2 as a result:

- **Part 1.a:** on one hand, we decided to set up a pre-production environment, not in the organization but in our own laboratory, by means of student<sup>2</sup> groups that were evaluating

<sup>2</sup>final project in a five year Computer Science degree

the process engines for BPMN2, XPD L and BPEL, as described in chapter 9. To do that, the BP model from the HGCR was given to the groups to be implemented and executed in the process engines selected (Activiti, Bonita and Intalio respectively), and registering data from the BP cases execution and exporting it to be loaded in the ProM framework. This will cover the steps 1 and 5 in the red circles, and steps 2 and 3 in purple squares.

- **Part 1.b:** on the other hand, the public enterprise responsible for telecommunications in Uruguay, ANTEL<sup>3</sup>, which was working in a project with the COAL Research Group, which this author belongs to, was attracted to the BPSOM proposal including the automatic generation of SoaML service models from BPMN2 models. After several meetings with the project leader, a BP model was selected to be prototyped using BPSOM with the automatic generation of services and the Eclipse MINERVA design distribution to support the defined activities. This will cover the steps 2, 3 and 4 in the red circles.
- **Part 2:** Based on the previous definitions, the second part of the case study was also affected, as the execution of the prototype was not being carried out in the context of the HGCR by the IT people but in our laboratory, and we were not able to execute a considerable number of BP cases to gather the needed data. This led us to decide to simulate the BP model to gather enough data, by using the CPNTools and information given to us from the HGCR quality group and M.D. experts in the BP. CPNTools allow us to register data from the simulation in MXML format to be loaded into the ProM plug-in we have developed to analyze the measurement results. This will cover the steps 6 and 7 in the red circles, and steps 1 and 4 in purple squares, as initially defined for Part 2.

In Figure 10.2 the final definition and coverage of the parts defined for the case study are shown in the MINERVA method of work. Steps from 1 to 5 in the red circles and steps 2 and 3 in the purple squares will be covered by the first part (Part 1.a and Part 1.b); steps 6 and 7 in the red circles and steps 1 and 4 in the purple squares will be covered by the second part (Part 2), as originally defined.

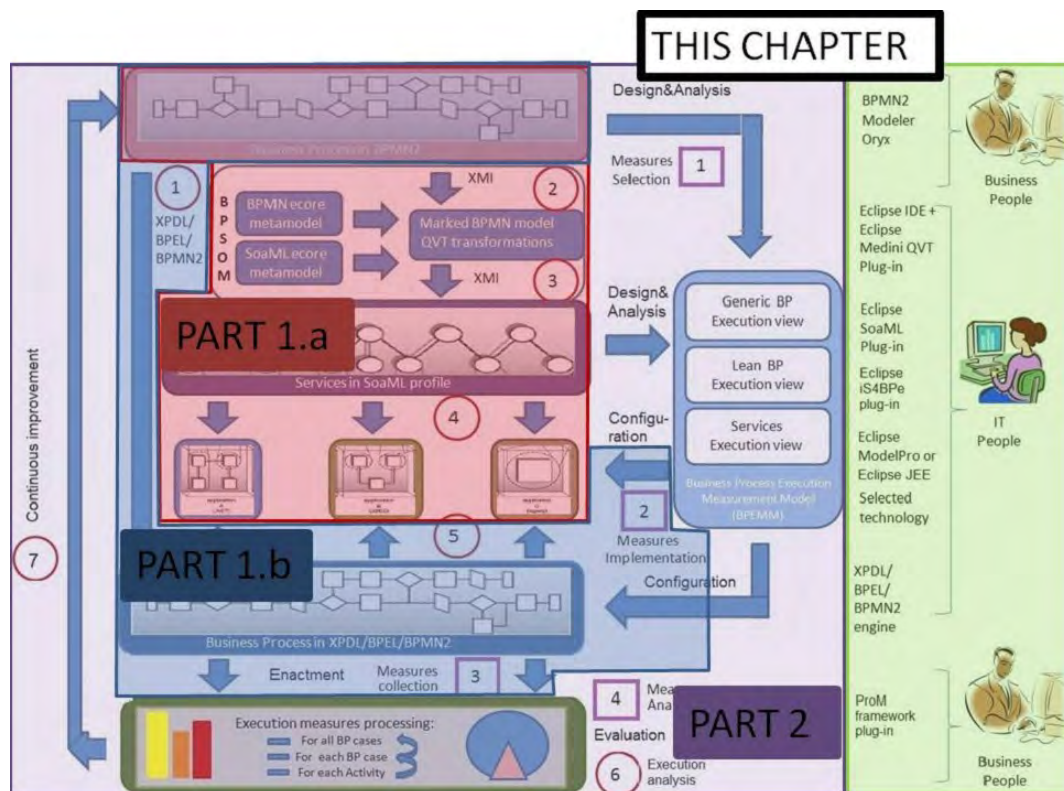


Figure 10.2.: MINERVA method of work coverage by the two case studies finally defined

<sup>3</sup><http://www.antel.com.uy/antel/>

The definitions presented led to the validation of MINERVA by means of two case studies with partial scope, instead of one, defined as: a case study for the validation of BPSOM carried out in the context of ANTEL corresponding to Part 1.b, described in section 10.3, and a case study for the validation of BPCIP carried out in the context of the HGCR and our laboratory corresponding to Part 1.a and Part 2, described in section 10.4. Although the case studies finally defined did not assess a single pass throughout the MINERVA framework we believe that in this way we are able to assess the feasibility of the different proposals integrated in MINERVA to provide insight on its application.

## 10.2. Empirical validation of QVT transformations

In this section the empirical validation of the defined QVT transformations is presented, for which we have followed the definition, procedures and guides in [Wohlin et al., 2000, Juristo and Moreno, 2001, Kitchenham et al., 2001]. The experiment carried out aimed to validate the Suitability (sub-characteristic of Functionality) and Understandability (sub-characteristic of Usability) as defined in ISO 9126 [ISO, 2001], of the QVT transformations defined and the SoaML service models generated by means of the transformations from BPMN2 models.

It was carried out thanks to the participation of several Computer Science Engineers from the University of Castilla - La Mancha, Spain and the University of the Republic, Uruguay which evaluated the service design we propose. For the statistical analysis of the data and carrying out the hypothesis testing we have used the statistical package SPSS v. 17.0<sup>4</sup>. As the replicability principle for empirical research establishes [Basili et al., 1999], the information on the experiment is given in detail below including its definition, material, method, analysis and interpretation of the results.

### 10.2.1. Problem definition

The aim of the experiment was to assess whether the QVT transformations we have defined provide software designers with a service design in the form of SoaML service models that corresponds to what they expected for realizing BP models with services, so they can use them in their development of services. The research question we want to answer is stated as:

*Do the QVT transformations defined between BPMN2 and SoaML models provide software engineers with service models that are appropriate to what they expect when modeling themselves, as well as usable as services design in their development of services from BP models?*

In terms of the Goal, Question, Metrics (GQM) [Basili, 1992] template as suggested by [Briand et al., 2002, Lott and Rombach, 1996] for the definition of experiments, this is defined as shown in Table 10.1.

**Table 10.1.:** Definition of the experiment in GQM

Analyze	QVT transformations between BPMN2 and SoaML models
with the purpose of	evaluating
with respect to	the Functionality of the transformations and the Usability of the generated SoaML models
from the point of view of	software engineers
in the context of	designing services to realize business processes

<sup>4</sup><http://www-01.ibm.com/software/analytics/spss/>

## 10.2.2. Planning of the experiment

In the following subsections the elements defined in the experimental plan are described, and are summarized in Figure 10.3.

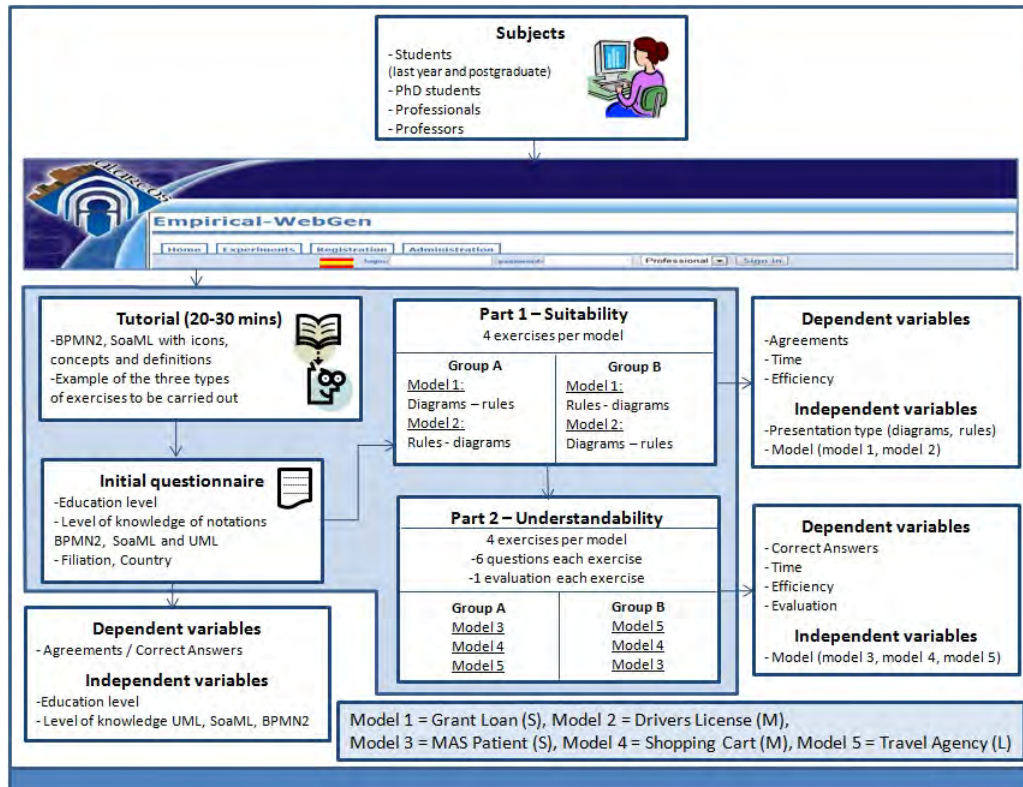


Figure 10.3.: Summary of the experimental plan

### 10.2.2.1. Context selection

The realization of the experiment requires the subjects to have studies of at least five years in Computer Science providing general knowledge on software modeling and design, and notations to specify such models as UML. We also want to cover different levels of education for the subjects, and with respect to students they had to be in the last year of a five-year computer science degree to guarantee the fulfilling of the previous requirements.

The student courses available at the University of Castilla - La Mancha at the time the experiment was to be conducted did not satisfy this condition, the PhD students satisfied this constraint but were only eight so we needed more subjects. Several subjects satisfying the conditions were available from the University of the Republic, Uruguay, but we could not perform the experiment on site, so we decided to carry it out online, as the University of Castilla - La Mancha has an application specifically developed to perform experiments online Empirical-WebGen [Alarcos, 2006]. So by email we asked several people satisfying the conditions defined to perform the experiment twenty one of whom actually did it.

The BP models used in the exercises were selected from real cases and well known examples used in university courses and from real life, although adapted to the needs of the experiment. The five BP models used for the experiment were: Grant Loan from a Bank, Major Ambulatory Surgery (MAS) from a Hospital, Drivers License from a Drivers License Office, Shopping Cart from a Selling Company Web site and Travel Booking from a Travel Agency. Each BP model has a different complexity associated, which is defined according to the number of services that will be generated from it, with S = small (4 services), M = medium (seven services) and L = large (eleven services).

### 10.2.2.2. Subjects selection

The selection of subjects was based on the satisfaction of the two main characteristics defined: that they should have a minimum of a five-year degree in Computer Science (or be studying the last year of a five-year degree), and have knowledge (for example from courses in the degree) and/or professional experience in software modeling. The total number of subjects that carried out the online experiment was twenty one.

They were not required to have a specific level of knowledge in the notations UML and SoaML profile for services design, nor on BPMN2 for BP modeling, although in a preliminary questionnaire we asked the subjects to indicate their level of knowledge in each of the notations, as well as their educational level, filiation and country. The level of knowledge for each notation was asked to be answered on a predefined scale from 1 to 5, being 1 - Very low, 2 - Low, 3 - Medium, 4 - High and 5 - Very high.

### 10.2.2.3. Variables selection

The quality characteristics to be evaluated in order to achieve the defined objective of the experiment were selected as defined in ISO 9126 [ISO, 2001](superseded by ISO/IEC 25000 [ISO, 2005-2011]):

- **Functionality:** “The capability of the software product to provide functions which meet stated and implied needs when used under specified conditions. The functions satisfy the formulated or supposed conditions”. The sub-characteristics of functionality are: Suitability, Accuracy, Interoperability, Security and Functionality Compliance.
- **Usability:** "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use". The sub-characteristics of usability are: Understandability, Learnability, Operability, Attractiveness and Usability Compliance.

The dependent variables for the experiment correspond to the sub-characteristic **Suitability** of Functionality and **Understandability** of Usability, which were measured using the variables explained below. For Suitability we count the agreements with the option corresponding to our proposal for deriving services from BPs by means of the QVT transformations we are validating, the time each subject takes to answer each exercise, and the efficiency in performing each exercise calculated as the number of agreements / time incurred.

For Understandability we count the correct answers for each of the six questions in each exercise about the services design generated by means of the QVT transformations, as well as the time each subject takes to answer each exercise, and the efficiency calculated in the same way as before but with the variable correct answers instead of agreements, along with the evaluation we asked for about the complexity of the SoaML diagram generated for each exercise. The summary of dependent variables defined to measure each sub-characteristic is shown in Table 10.2.

**Table 10.2.:** Measures for the selected dependent variables

	Agreements	Correct answers	Time	Efficiency	Evaluation
Suitability	X	–	X	X	–
Understandability	–	X	X	X	X

### 10.2.2.4. Formulation of hypothesis

The answers to the research question for each sub-characteristic of Suitability and Understandability were provided directly from the percentage of agreements and correct answers for each part of the experiment, as we were not comparing our method with other. That being the case, we



can not expect results of the type “using this is better than using that”; the hypothesis for each sub-characteristic were thus defined to assess the threats to the validity of the experiment related to the influence of the variation of the independent variables on the dependent variable results.

For the Suitability sub-characteristic we have defined two independent variables: the presentation of the QVT transformations as generated SoaML diagrams or as textual correspondence rules, and the complexity of the BP model with respect to the number of services in the design it yields to; and for Understandability the complexity of the BP model is defined, as shown in Table 10.3.

**Table 10.3.:** Central hypotheses for Suitability and Understandability assessment

Dependent variable	Measured by	Hypothesis	Independent variables
Suitability	Agreements Time Efficiency	H0.a = the presentation of QVT transformations for the generation of SoaML service models from BPMN2 as diagrams or textual correspondence rules has no effect on the Suitability results H1.a = $\neg$ H0.a	Type of presentation (diagram, textual rules)
		H0.b = the Complexity of the BP model has no effect on the Suitability of the QVT transformations for the generation of SoaML service models from BPMN2 H1.b = $\neg$ H0.b	Complexity of the BP model
Understandability	Correct answers Time Efficiency Evaluation	H0.c = the complexity of the BP model has no effect on the understandability of the result of the QVT transformations for the generation of SoaML service models from BPMN2 models H1.c = $\neg$ H0.c	Complexity of the BP model

Based on the answers to the initial form in the experiment about education level and level of knowledge of UML, SoaML and BPMN2 notations that we have asked for in the initial questionnaire, we have also defined some complementary hypotheses to assess the influence of these variables on the results, as shown in Table 10.4.

**Table 10.4.:** Complementary hypotheses for Suitability and Understandability assessment

Dependent variable	Measured by	Hypothesis	Independent variables
Suitability / Understandability	Agreement/ Correct answers	H0.d = education has no effect on the Suitability/Understandability of the (result) QVT transformations for the generation of SoaML service models from BPMN2 H1.d = $\neg$ H0.d	Education level
		H0.e = UML knowledge has no effect on the Suitability/Understandability of the (result) QVT transformations for the generation of SoaML service models from BPMN2 H1.e = $\neg$ H0.e	UML knowledge level
		H0.f = SoaML knowledge has no effect on the Suitability/Understandability of the (result) QVT transformations for the generation of SoaML service models from BPMN2 H1.f = $\neg$ H0.f	SoaML knowledge level

Dependent variable	Measured by	Hypothesis	Independent variables
Suitability / Understandability	Agreements/ Correct answers	$H0.g = \text{BPMN2 knowledge has no effect on the Suitability/Understandability of the (result) QVT transformations for the generation of SoaML service models from BPMN2}$ $H1.g = \neg H0.g$	BPMN2 knowledge level

### 10.2.2.5. Design of the experiment

The experiment was defined in two parts: the first one corresponding to the dependent variable Suitability (from now on Part 1) and the second one corresponding to the dependent variable Understandability (from now on Part 2).

The subjects were split randomly into two groups called Group A and Group B by assigning each subject accepting to perform the experiment to a different group sequentially when the answer email was received, starting with Group A, so that the assignation of subjects to groups was balanced. In this way, eleven subjects were placed in Group A and ten subjects were placed in Group B.

#### Part 1 - Suitability

Two BP models were used: the Grant Loan from a Bank corresponding to Model 1, and the Drivers License from a Drivers Office corresponding to Model 2; the same diagrams and textual correspondence rules were given for each Group, but in different order. Group A performed the first assignment on the Grant Loan model answering for each SoaML diagram exercise in the first place the diagram questions and secondly the textual correspondence rules; the second assignment was performed on the Drivers License model answering for each SoaML diagram exercise the textual correspondence rules questions in the first place and then, secondly the diagrams.

Group B did the opposite, for the Grant Loan model the textual correspondence rules questions were answered in the first place, and then the diagram ones for each SoaML diagram exercise, and for the Drivers License model firstly the diagram questions were answered and then, secondly the textual correspondence rules. As mentioned the complexity of BP models was defined based on the quantity of services that can be derived to realize them, with the value for the Grant Loan small (four services) and for the Drivers License medium (seven services). Each exercise presents both the BPMN2 model and the corresponding SoaML diagrams or textual correspondence rules for the transformation being assessed. The design for the experiment Part 1 - Suitability corresponding to a 2x2 factorial is shown in Table 10.6.

**Table 10.5.:** Design of the experiment Part 1 - Suitability

Suitability	BP Model	
	Grant Loan	Drivers License
Diagrams - Rules	Group A	Group B
Rules - Diagrams	Group B	Group A

#### Part 2 - Understandability

Three additional BP models were used: the Patient MAS from a Hospital corresponding to Model 3, the Shopping Cart from a selling Company Web Site corresponding to Model 4, and the Travel Booking from a Travel Agency corresponding to Model 5. The complexity of the models as defined before were: for the Patient MAS small (four services), for the Shopping Cart medium (seven services) and for the Travel Booking large (eleven services). For each BP model four exercises making up the SoaML diagrams generated by means of the QVT transformations were provided

along with the corresponding BPMN2 model, and for each diagram six questions were asked about the meaning of the elements in the SoaML diagram, to be answered by selecting True or False. The order of the six questions was randomized for each individual answering the exercises.

For each exercise corresponding to a SoaML diagram, the subjects were also asked to evaluate the complexity of the SoaML model based on a predefined scale from 1 to 5, being 1 - Very simple, 2 - Relatively simple, 3 - Normal, 4 - Relatively difficult and 5 - Very difficult. It is worth noting that we are not asking for an evaluation of the complexity of the BP model but of the SoaML model generated using the QVT transformations, as we are assessing the understandability of elements for each SoaML diagram. The design for the experiment Part 2 - Understandability corresponding to a within-subject design in which all subjects had to answer all the tests is shown in Table 10.6, with the order in which each Model was presented.

**Table 10.6.:** Design of the experiment Part 2 - Understandability

Understandability	BP Model		
	Patient MAS	Shopping Cart	Travel Booking
Group A	1	2	3
Group B	3	2	1

It should be remarked that because the experiment is online and due to the particular design of the application once an exercise is finished by the user, there is no going back to review it or to change the answer, so the exercises can not be compared (which is what we wanted). This applied to both parts of the experiment.

#### 10.2.2.6. Experimental materials

Both groups were given three different materials to perform the experiment: a tutorial on the notations BPMN2 and SoaML and one example for each of the exercises they will perform: Part 1 of the experiment for evaluating the Suitability sub-characteristic and Part 2 of the experiment for evaluating the Understandability sub-characteristic, described below. The complete material for the experiment is provided in Appendix D in Spanish as this is the language in which the experiment was carried out, although some parts have been translated for inclusion in this chapter.

##### Tutorial

The tutorial consisted of the presentation of BPMN2 and SoaML notations and their main elements, as well as a third part containing an example of an exercise for each of the two parts of the experiment.

##### Part 1

Part 1 of the experiment which evaluates the Suitability of the QVT transformations, consisted in performing four exercises for each of the two BP models provided, where in each exercise a different SoaML diagram for the design of services to realize the BP is given: Services Architecture, Services Interfaces, Services Contracts, and Participants&Services.

Each exercise also consisted in two parts: one where the design options were provided only with the SoaML diagrams and only one had to be selected, and another where the design options were provided as textual correspondence rules from which only one also had to be selected. The order of the diagrams - rules was different for each BP model for each given group A and B, as described above. An example of the exercises from this Part 1 for the SoaML Services Architecture diagram in graphical form is shown in Figure 10.4 and in textual form in Figure 10.5.

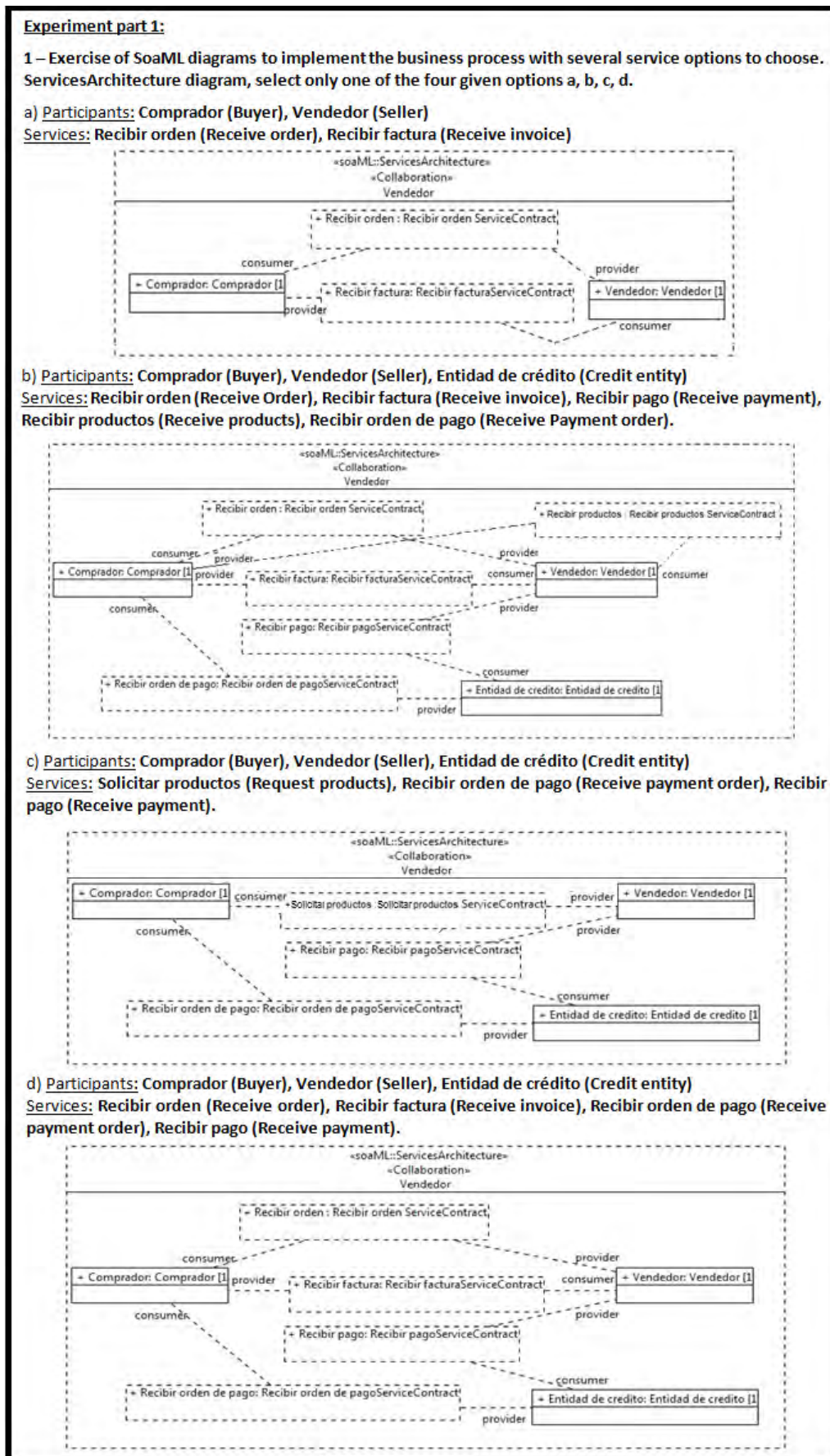


Figure 10.4.: Part 1 example of diagrams exercise for Suitability

**2 – Exercise of rules definition for services from business process with several options of services to choose. Select one of the four given options a, b, c, d.**

The Services Architecture corresponds to:

a) the collaborative process “Venta de productos” (Sale of products) and includes:

- participants: Comprador (Buyer), Vendedor (Seller), Entidad de crédito (Credit entity)
- services corresponding to activities and conceptual merge of activities involved in a message flow between two different processes:
  - Request products (provider Seller, consumer Buyer)
  - Receive payment order (provider Buyer, Consumer Credit entity)
  - Receive payment (provider Seller, consumer Credit entity)

b) the collaborative process “Venta de productos” (Sale of products) and includes:

- participants: Comprador (Buyer), Vendedor (Seller), Entidad de crédito (Credit entity)
- services corresponding to activities involved in a message flow between two different processes:
  - Receive order (provider Seller, consumer Buyer)
  - Receive invoice (provider Buyer, consumer Seller)
  - Receive products (provider Buyer, consumer Seller)
  - Receive payment order (provider Credit entity, consumer Buyer)
  - Receive payment (provider Seller, consumer Credit entity)

c) the collaborative process “Venta de productos” (Sale of products) and includes:

- participants: Comprador (Buyer), Vendedor (Seller), Entidad de crédito (Credit entity)
- services corresponding to activities involved in a message flow between two different processes:
  - Receive order (provider Seller, consumer Buyer)
  - Receive invoice (provider Buyer, consumer Seller)
  - Receive payment order (provider Credit entity, consumer Buyer)
  - Receive payment (provider Seller, consumer Credit entity)

d) the processes Buyer and Seller part of the collaborative process “Venta de productos” (Sale of products) and includes:

- participants: Comprador (Buyer), Vendedor (Seller)
- services corresponding to activities involved in a message flow between two different processes:
  - Receive order (provider Seller, consumer Buyer)
  - Receive invoice (provider Buyer, consumer Seller)

Figure 10.5.: Part 1 example of textual correspondence rules exercise for Suitability

## Part 2

Part 2 of the experiment which evaluates the Understandability of the SoaML diagrams resulting from the QVT transformations, consisted in performing four exercises for each of the three BP models provided. Each exercise presented a SoaML diagram for which six questions had to be answered with True or False, about elements in the diagram.

In this part, for each BP model only the generated diagrams from the QVT transformations were provided, as we were evaluating the service models generated. The order in which the BP models were presented was different for each given group A and B, as described above, and the questions for each exercise were randomized for each subject. After finishing each exercise the evaluation of the SoaML diagram complexity was also asked for, in the defined scale from 1 to 5. In Figure 10.6 an example for the Services Architecture diagram is shown.

**Experiment part 2:**

**1 – Exercise of SoaML diagrams to identify different elements in the given diagram answering with True/False. Answer the 6 questions associated with the diagram.**

1.1) The participant Seller provides the services “Recibir orden” (Receive order), “Recibir factura” (Receive invoice), “Recibir pago” (Receive payment), “Recibir orden de pago” (Receive payment order).  
T F

1.2) The provider role in the service contract of the service “Recibir orden de pago” (Receive payment order) is played by the participant Credit entity.  
T F

1.3) The participant Patient plays the consumer role defined in the service contract of the service “Recibir factura” (Receive invoice).  
T F

1.4) The service contract for the service “Recibir orden” (Receive order) defines the interaction between the participants Buyer and Seller with the service, with the roles consumer and provider respectively.  
T F

1.5) The roles consumer and provider defined for the service “Recibir factura” (Receive invoice) are played by the participants Buyer and Seller respectively.  
T F

1.6) The participant Credit entity interacts with the participant Buyer through the service “Recibir pago” (Receive payment) playing the role of the consumer.  
T F

**2 – Exercise to assess the complexity that you think the SoaML diagram presents, in the scale from 1 to 5 defined. Asses the complexity of the ServicesArchitecture SoaML model presented in Exercise 1:**

1 – Very simple  2 – Relatively simple  3 – Normal  4 – Relatively difficult  5 – Very difficult

Figure 10.6.: Part 2 example of exercises for Understandability

### 10.2.2.7. Assessment of validity

The validity threats that could affect the experiment were analyzed as part of the planning of the experiment and are described below:

- Construct validity:
  - The measures agreements/correct answers, time and efficiency selected are normally used in Software Engineering empirical research to measure dependent variables.
  - As the experiment was carried out online the times were store automatically for each exercise once the subject had finished the previous one, so there was no possibility of wrong or missing start or finish time in the exercises. The threat that the times might not be real if the person left in the middle of an exercise was reduced by the time-out for idle time the web application has, set on 20 minutes, and by telling the subjects that once started they had to finish the part of the experiment because the time-out would reset all the exercises

they had done until that moment. The subjects had also agreed to perform the experiment so the level of commitment was high.

- Internal validity:

- Persistence validity: the majority of the subjects had not performed an experiment before and although five of them the PhD students of the UCLM had, we believe this did not affect the overall results.

- Knowledge of the BP models or domains: the knowledge of the domains had not affected the results as for each exercise a different BP model from a different domain was provided, with all domains known by everyday terms (Loan in a Bank, a procedure in a Hospital, license drivers request, etc.) to avoid the effect of having to understand the problem.

- Fatigue effects: to avoid these effects each part was an hour long and the experiment could be performed in the moment the subject chooses and not on a fixed day where subjects could be tired from other activities.

- Subjects motivation: subjects were asked to agree to perform the experiment so no one was forced to perform it, explaining to them the help it would be for this thesis work.

- Plagiarism between subjects: this was not controlled but the subjects were committed to performing the experiment and we believe there had not been plagiarism.

- External validity:

- The subjects were selected on the basis of the restrictions defined and we had a heterogeneous sample including PhD students, Students of postgraduate and graduate (last year) degrees, Professors and Professionals, although it was too small to be able to generalize the results. A replication is planned to be able to confirm the obtained results.

- Conclusions validity:

- The complete set of data consisted of 336 answers for part 1 of the experiment, 1512 answers for part 2 of the experiment and 252 answers for part 2 evaluation of the complexity. We thus believe this threat is not present as the data set is considerable.

### 10.2.3. Operation of the experiment

In this section the tasks performed prior to and during the execution of the experiment are described.

#### 10.2.3.1. Preparation

Once the experimental material was available online for the two versions (Group A, Group B) a Computer Science Engineer with software modeling knowledge carried out the experiment so we were able to evaluate the time that each Part involves (which was around an hour each as expected) as well as to correct some mistakes in the material.

#### 10.2.3.2. Execution

As mentioned before several subjects were asked by email to perform the experiment online, providing them with key information about it such as that it consists of two separate parts that will take around an hour each and that they had to perform part 1 in the first place and then part 2. They were also told that each part was independent from the other and that they could perform them at different times while respecting the order, as well as that each part had to be finished once started. They were given a week to perform the experiment so they could evaluate the deadline before committing themselves to the task, the WebGen and tutorial links, the Group assignation (A or B) and respective user and password.

The data corresponding to the answers of the subjects performing the experiment was stored in the data base of the application, which could be extracted afterwards in several different formats such as excel sheet or pdf. During the week given to the subjects to perform the experiment this was monitored by means of a function in the web application that allows the users that had performed it to be seen, along with the state of each part of the experiment, which was useful to send a reminder before the deadline.

### 10.2.3.3. Data validation

The web application had controls to ensure that each exercise and each question were answered as defined, preventing the subjects from going further ahead in the exercises until they had finished the current one. The monitor option of the web application also allowed us to check each subject and each part of the experiment for missing data, which was not the case and all twenty one executions of the experiment were valid.

## 10.2.4. Data analysis and interpretation of results

In this section the analysis of the data collected from the experiment and its interpretation is presented. From the exercises of Part 1 the total answers processed were 336, and from the exercises of Part 2 the total number of answers were 1512 for the questions about the SoaML diagrams and 252 for the evaluation of the complexity of the SoaML diagrams. In the first place the descriptive statistics are presented, followed by the hypothesis testing in which the interpretation of results is also presented.

### 10.2.4.1. Descriptive statistics

After collecting the data, the answers were reviewed and all of them were considered valid and the descriptive statistics were obtained, which are shown in Table 10.7 for both sub-characteristics Suitability and Understandability.

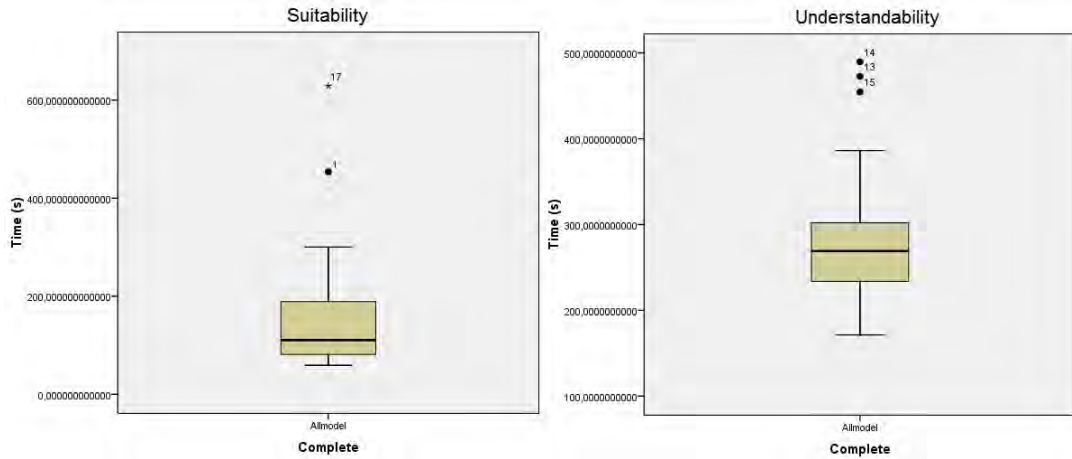
**Table 10.7.:** Descriptive statistics for Suitability and Understandability

Dependent variables	Measured by	Mean	std. deviation
Suitability	Agreements (%)	82,14	38,36
	Time (s)	154,07	120,61
	Efficiency	0,764	0,348
Understandability	Correct answers (%)	75,38	12,56
	Time (s)	290,83	85,92
	Efficiency	0,277	0,081
	Evaluation	Median = 3	0,472

The percentage of Agreements for Suitability indicates that the SoaML service design solution proposed by means of the QVT transformations is appropriate for the service design the users expect in an average of 82% of the cases. For the Understandability of the generated SoaML service models the percentage or Correct Answers indicates that users understand them in an average of 75% of the cases. These percentages then, are the answers to our research question in a positive way, indicating that the solution proposed for designing services from BP models can be used and understood by software engineers in those high percentages.

With respect to the average time incurred in answering each exercise, it can be seen that for Part 2 it is twice the time for Part 1, with the percentage of correct answers less than the agreements, the efficiency for Part 2 is almost the third of the efficiency for Part 1. The graphical representation as a box plot is shown for the average time in Figure 10.7.





**Figure 10.7.:** Average times for Suitability and Understandability

It can be seen in Figure 10.7 that there are two outliers for the Suitability times which correspond to the average times for the first exercise for Group A and Group B respectively. This could be due to the fact that as this is the first exercise each person faces, it took more time to understand what was required, after this exercise the times are stabilized as shown in the diagram. For Understandability there are also three outliers which correspond to the average times of the three first exercises of version B (evaluating Model 5). This could be due to their being the first exercises for Part 2 on a model which presents the highest complexity of the three models provided.

For Suitability, these results can be further broken down into results per presentation type, that is diagrams and textual correspondence rules, where the order of exercises with diagrams and rules is changed for each BP model and for each Group A and B, as are the results for each Model, Model 1 being of small complexity and Model 2 being of medium complexity. For Understandability the results can be further broken down for each Model, Model 3 being of small complexity, Model 4 of medium complexity and Model 5 of large complexity. The results are shown in Table 10.8 for Suitability and in Table 10.9 for Understandability.

**Table 10.8.:** Results for Suitability per presentation type and model

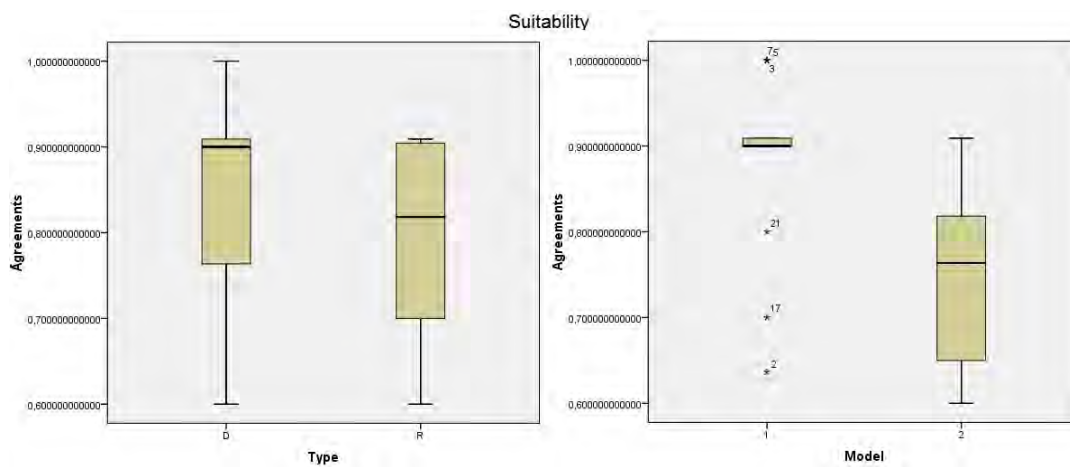
Dependent variable	Independent variables							
	Presentation type				Complexity of BP Model			
Suitability	Diagrams		Text Rules		Model 1		Model 2	
Measured by	Mean	std.dev.	Mean	std.dev.	Mean	std.dev.	Mean	std.dev.
Agreements (%)	84,83	13,01	79,09	11,30	88,58	31,8	75,17	43,1
Time (s)	135,11	156,67	173,33	203,07	195,69	231,36	112,75	97,61
Efficiency	0,832	0,374	0,625	0,279	0,698	0,387	0,798	0,293

The percentage of agreements is higher for Diagrams (84,83%) than for Rules (79,09%), times for Diagrams (135,11s) are lower than times for Rules (173,33s), so for Diagrams (0,832) efficiency is greater than for Rules (0,625). The percentage of agreements is higher for Model 1 (88,58%) than for Model 2 (75,17%), and times for Model 1 (195,69s) are higher than for Model 2 (112,75s), and results for efficiency are similar but higher for Model 2 (0,798) than for Model 1 (0,698). Differences between times by model could be explained by the fact that Model 1 was the first model in the exercises so for Model 2 the way in which the exercises had to be solved was already known, as was how the online application worked.

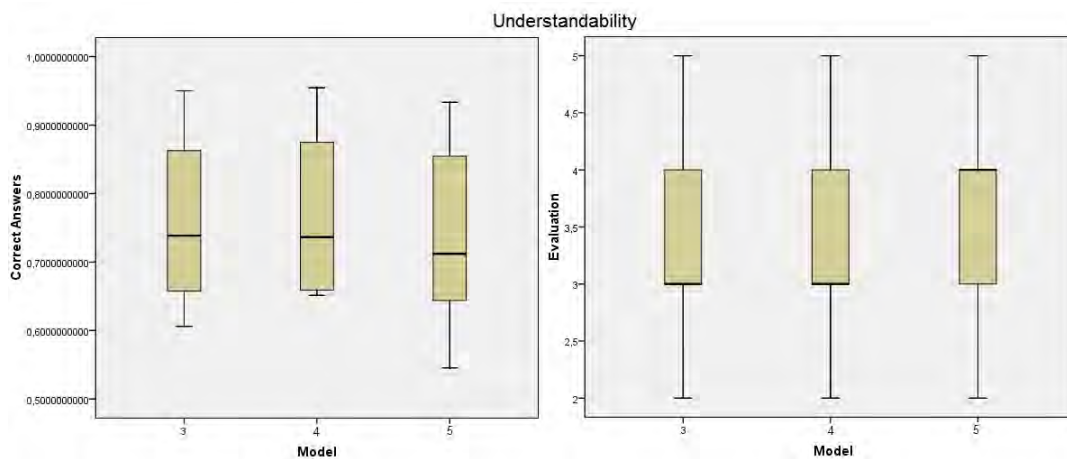
**Table 10.9.:** Results for Understandability per model

Dependent variable	Independent variable = Complexity of BP Model					
	Model 3		Model 4		Model 5	
Understandability	Mean	std.dev.	Mean	std.dev.	Mean	std.dev.
Measured by						
Correct answers (%)	75,93	12,45	76,46	13,03	73,75	13,76
Time (s)	259,63	66,16	263,29	26,78	349,56	115,07
Efficiency	0,308	0,091	0,293	0,058	0,229	0,076
Evaluation	Median = 3	0,638	Median = 3	0,685	Median = 4	0,789

In Figure 10.8 the diagram form of percentage of Agreements per presentation type (diagrams = D, rules = R) and per model (Model 1 = 1, Model 2 = 2) is shown as a box plot; and in Figure 10.9 the diagram form as a box plot is presented for percentages of Correct Answers per model (Model 3 = 3, Model 4 = 4, Model 5 = 5) and for the evaluation of the complexity of the SoaML diagrams.

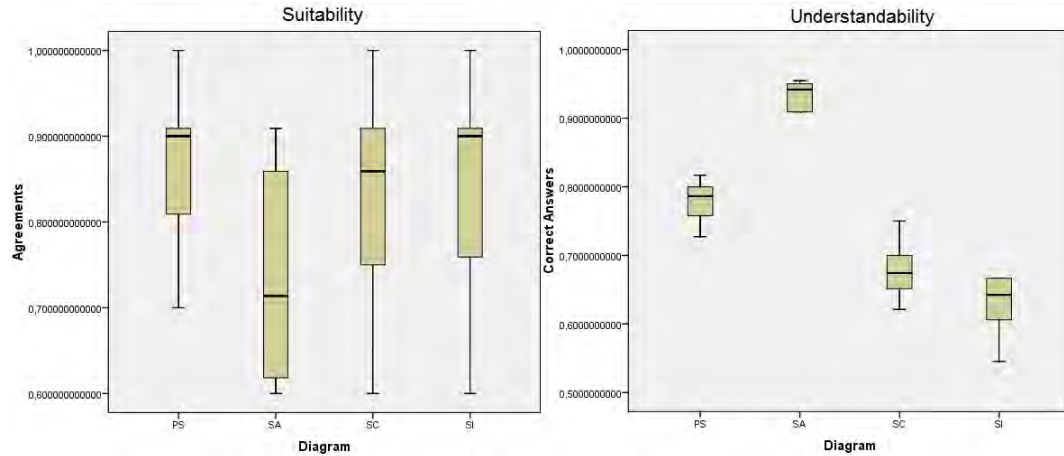


**Figure 10.8.:** Agreements per presentation type and model for Suitability



**Figure 10.9.:** Correct answers and evaluation per model for Understandability

Another statistic we were interested in is the percentage of Agreements and Correct Answers for each of the four SoaML diagrams presented for each BP model in each corresponding exercise, which are shown in graphical form as box plots in Figure 10.10.



**Figure 10.10.:** Agreements and Correct Answers per SoaML diagram

It can be seen that all of the values are above 60%; for the Suitability sub-characteristic the higher percentages of Agreements correspond to the Services Interfaces diagram and the Participants and Services diagram, and the lower ones are with respect to the ServiceArchitecture diagram, which was the first diagram presented in the exercises. On the other hand, for the Understandability sub-characteristic the highest percentage corresponds to the Service Architecture diagram and the lowest to the Services Interfaces diagram.

We also investigate the effect on the results of the Agreements and Correct Answers for the variables that characterize the subjects: education level and level of knowledge of the notations UML, SoaML and BPMN2, whose descriptive statistics are shown in Table 10.8.

**Table 10.10.:** Percentages of agreements for education and notations knowledge level

Indep. variables	Dep. variables	Agreements (%)		Correct Answers (%)	
	Value	Mean	std. dev.	Mean	std. dev.
Education	1 - Student (LY)	1	–	69	46,4
	2 - Student (P)	1	–	69	46,4
	3 - PhD student	84	38,6	75	43,2
	4 - Professor	98	14,4	67	47,1
	5 - Professional	68	46,9	81	39,6
UML	1 - Very low	1	–	71	45,8
	2 - Low	85	35,7	77	42,3
	3 - Medium	74	43,7	75	43,2
	4 - High	97	17,7	72	44,9
	5 - Very high	94	24,5	77	42,0
SoaML	1 - Very low	78	41,7	75	43,2
	2 - Low	90	30,7	74	44,0
	3 - Medium	1	–	85	33,2
BPMN2 level	1 - Very low	74	43,7	73	44,3
	2 - Low	98	15,7	83	37,3
	3 - Medium	75	43,8	69	46,2
	4 - High	97	17,7	75	43,5

#### 10.2.4.2. Hypothesis testing

After analyzing the descriptive statistics for both of the sub-characteristics of Suitability and Understandability the hypothesis testing was carried out to see whether the differences observed were statistically significant. For the sub-characteristic Suitability an ANOVA test with significance level  $\alpha = 0,05$  was selected and for the sub-characteristic Understandability a covariance analysis

with the Pearson correlation coefficient with significance level  $\alpha = 0,05$  was selected. In Table 10.11 the results for Suitability and Understandability sub-characteristics are shown.

**Table 10.11.:** Significance levels for presentation type and Model

Dependent variables	Measured by	Independent variables	
		Presentation Type	Complexity of BP Model
		p-value	p-value
Suitability	Agreements	0,150	<b>0,002</b>
	Time	<b>0,048</b>	<b>0,000</b>
	Efficiency	0,080	0,280
Understandability	Measured by	Complexity of BP Model	
		Corr.Coeff.	p-value
	Correct answers	-0,021	0,422
	Time	0,436	<b>0,033</b>
	Efficiency	-0,407	<b>0,049</b>
Evaluation	0,279	<b>0,000</b>	

As can be seen in Table 10.11, regarding the Suitability sub-characteristic no significant results were found for the Presentation Type (diagram, rules) on the Agreements or for either the Presentation Type or the Model on the Efficiency. Significant results were found for the Agreements and Time dependent variables, in the first case by the Complexity of the BP Model, and in the second case by the Complexity of the BP Model and the Presentation Type. Due to this fact, the null hypothesis H0.b for Agreements and Complexity of BP Model variable could be rejected, and the null hypothesis H0.a for Time and Presentation Type and Complexity of BP Model could also be rejected. For the Understandability sub-characteristic no significant results were found for the Correct Answers for the Complexity of the BP Model, but significant results were found for the Time, Efficiency and Evaluation variables so the null hypothesis H0.c could be rejected for each, concluding that the Complexity of the Model affected these variables.

For the Suitability sub-characteristic, it can be concluded that subjects agreed more with the generated solution for small models, as it was expected that the complexity of the BP model will influence the overall view of the services design. It can also be concluded that the Presentation Type of diagrams helped the subjects in answering in less time than the textual correspondence rules presentation. The Model also influenced the time of answering but in the opposite direction than expected, as Model 1 took more time than Model 2 which is defined as more complex. This could be due to Model 1 being the first model to be answered so although Model 2 was more complex the subjects were already comfortable with the exercises so the time incurred did not include the understanding of these. All of this lead us to plan to do a replication improving the design of the experiment to test this again.

For the Understandability sub-characteristic, it can be concluded that the different models have no effect on the correct answers, looking at the descriptive statistics it can be seen that they present similar percentages of Correct Answers for the three models. In the case of the time variable, it can be concluded that as the complexity of the model grows it takes more time for the subjects to understand them, as was expected. The efficiency for each model is thus inverse to the times, as the percentage of Correct Answers were similar, meaning that as the complexity of the models grows the subjects were less efficient in understanding them. For the Evaluation variable although the median for the associated SoaML diagrams is the same for the small and the medium BP models, the highest evaluation corresponds to the large BP model, so as complexity grows in the BP model the subjects evaluate the associated SoaML diagram also as more complex, which was as expected.

For the variables corresponding to education level and knowledge level of the notations UML, SoaML and BPMN2 for the Agreements and Correct answers variables, a covariance analysis with the Spearman correlation coefficient with significance level  $\alpha = 0,05$  was selected, obtaining the results shown in Table 10.12.

**Table 10.12.:** Significance levels for education, UML, SoaML and BPMN2

Independent variables	Dependent variables			
	Suitability		Understandability	
	Agreements		Correct answers	
	Corr.Coeff.	p-value	Corr.Coeff	p-value
Education	-0,221	<b>0,000</b>	0,063	<b>0,014</b>
UML	0,067	0,220	0,005	0,861
SoaML	0,171	<b>0,002</b>	0,009	0,716
BPMN2	0,176	<b>0,001</b>	0,021	0,415

Significative results were found for Education in both sub-characteristics Suitability and Understandability so the H0.d null hypothesis could be rejected concluding that the educational level has an effect on the results of the Agreements and Correct Answers subjects gave in the experiment. Although these results were significative there is no clear relation between the level of education and the percentage of agreements or correct answers, so we plan to replicate the experiment taking this variable into account to further asses its influence.

The UML level of knowledge does not present significative results for any of the sub-characteristics, but the SoaML and BPMN2 level of knowledge also present significative results but only for the Agreements variable, so the H0.f null hypothesis is rejected for SoaML and the H0.g null hypothesis is rejected for BPMN2 for the Agreements variable. Hence it can be concluded that as all the subjects had knowledge (different levels in each case) of UML but only a few had any knowledge of SoaML and BPMN2, the variations on these two variables were the ones affecting the results for the Agreements variable.

### 10.2.5. Presentation and dissemination

Based on the analysis and interpretation performed on the results of the experiment we could sense a trend towards being able to say that the QVT transformations defined to automatically generate SoaML service models from BPMN2 models are suitable and usable by software engineers. We can not generalize the results, however, as we need to replicate the experiment to take into account the insights provided by this execution.

The complete definition and results of the experiment are written in a technical report and will be made available in the Web site of this thesis<sup>5</sup>.

## 10.3. Empirical validation of BPSOM

To asses the applicability of BPSOM and the automatic generation of SoaML service models from BPMN2 models a case study was carried out in a real organization, corresponding to the definition of Part 1.b presented in section 10.1. Our aim was to see whether the methodology guide, the QVT transformations and the set of tools to support the activities defined are appropriate and useful for a software development project guided by BPs. From the point of view of the organization the needs to be satisfied include the explicit creation and use of models in all development stages, traceability between models, facilities to support development activities and support for increasing development productivity.

The case study was carried out in the public enterprise responsible for telecommunications in Uruguay, ANTEL <sup>6</sup>, which is currently undertaking new software development projects based on services to provide support to the use of cell phones as payment method, known as electronic wallet. ANTEL is the lead telecommunications provider in Uruguay with a monopoly on landlines, offering other services such as broadband and cell phones, and providing services to the whole country of around 3.500.000 inhabitants. It has about 6.000 employees all over the country, including its own

<sup>5</sup><http://alarcos.esi.uclm.es/MINERVA/>

<sup>6</sup><http://www.antel.com.uy/antel/>

IT department which is in charge of developing new commercial products, maintaining the existing ones, telecommunications, servers, etc.

The case study was carried out in the context of the electronic wallet development project from the IT division, for which a representative BP process was selected, and also as a final work for a postgraduate course which two members of the team were attending. In the following the definition, execution, data collection and analysis of the case study is presented based on the guides in Yin [2002], Brereton et al. [2008].

### 10.3.1. Background

A systematic review on the topic of BPM, SOC and MDD was carried out at the beginning of this thesis as presented in chapter 3, identifying as key element of the research the main principles for the integration of paradigms.

The main research question for this case study is defined as:

*Does MINERVA provide by means of BPSOM a useful proposal for carrying out service-oriented software developments from BPs in organizations ?*

Additional research questions derived from this are defined as:

- does BPSOM provide a better way to undertake this kind of development projects than the ones carried out previously in organizations?
- are QVT transformations defined to generate SoaML service models from BPMN2 models appropriate and useful to the software designers in the development process?
- is the tool support provided to carry out service-oriented development by means of BPSOM appropriate and useful to the software development team?

### 10.3.2. Design

The type of case study carried out was a single-case in a single organization and in a single project of the organization, corresponding to an holistic case [Yin, 2002]. The object of the study was the service-oriented development of a BP in the context of the electronic wallet project, being the unit of analysis the project itself. Several other sub-questions were defined based on the additional research questions for the specific organization and project:

- were the guides provided, the notation and tools selected for modeling BPs in the organization useful to perform the activities and generate artifacts in the development process?
- was the tool support provided by means of the Eclipse MINERVA design distribution useful for carrying out the activities and generating artifacts in the development process?
- were the QVT transformations for SoaML service models generation from BPMN2 models integrated in Eclipse MINERVA design distribution appropriate and useful in obtaining the service design desired in the development process?
- were the guides and tool support provided for navigating from SoaML service models to implementation and from BPMN2 models to executable BPMN2 models appropriate and useful in the development process?

The data collected to answer these and the previously defined research questions was obtained from the answers of the participants to a questionnaire that was designed to asses the development of the project in the organization. The questionnaire has five sections and each of them in turn, has between one and five questions to be answered in a defined scale of five values: 1 - Little, 2 - Some, 3 - Medium, 4 - Quite and 5 - Much (closed questions). There were also open questions in which the participants can write their point of view and comments.

### 10.3.3. Case selection

The electronic wallet project from the organization was selected for several reasons: it is currently a project under development in the organization in an iterative incremental way of work, and defining several modules that have to cooperate in order to provide the defined functionalities. It involves a team in the organization with which the research group COAL in Uruguay, to which this author belongs, was already working providing training on BP modeling and services development, so the learning curve was minimized.

The project leader was interested in applying new approaches to improve the development process so the case study has her sponsorship and active participation, with her also selecting the specific BP to be used in the case study. The organization has a real need to improve the use of models and documentation in its development processes and tools to support all the activities. They already have an infrastructure which includes one of the process engines we had evaluated, JBoss with jBPM, although they were open to using others such as Activiti.

### 10.3.4. Procedures and roles

The main roles applied in the case study were the Business Analyst, Software Architect, Analyst and Developer, as defined in BPSOM. As the project was an internal software project from the IT division, the people who had the knowledge of the business were also the software team, so the Business Analyst role was also performed by the team. The procedure to carry out the case study followed the BPSOM methodology with an introductory phase in which the Eclipse MINERVA design was given to the project team also providing several technical guides for the integrated plug-ins. The participants received training in BPMN2 (12 hours) including workflow patterns and best practices on modeling, and SoaML (3 hours) since they already knew UML.

This researcher acted as a consultant on MINERVA, BPSOM and Eclipse MINERVA design but all the activities and artifacts were carried out and generated by the software team themselves. In the first place the BP was modeled in BPMN2, marking the activities to be generated as services with the “ServiceTask” type, saved in BPMN2 format. It was loaded in the Eclipse MINERVA design, transformed into XMI format and the QVT transformations were executed to generate the SoaML service models. The SoaML plug-in was used to import the obtained XMI service model and visualize the diagrams generated.

After the application of BPSOM was finished as defined by the development team, the questionnaires were given to the members of the team participating in the case study.

### 10.3.5. Data collection

The collection of data was performed from the answers to the questionnaire given to the participants in the case study, using the scale given. Some open questions were defined in which they had to provide tools or characteristics assessed as answers, and also evaluate them using the scale, along with a final open question to be answered freely with comments and observations. The questionnaire produced is presented in Table 10.13:

**Table 10.13.:** Questionnaire to assess the use of BPSOM in the case study

Questions
<b>1 - About the guidelines provided for the modeling of BPs in the organization</b>
1.1 - The information and references provided to carry out the modeling were useful
1.2 - The language BPMN2 selected for BP modeling was useful to specify the BP model
1.3 - Name the tools used and their usefulness to support the activities defined in BPSOM
<b>2 - About the Eclipse MINERVA distribution</b>
2.1 - It was useful in supporting the activities defined in BPSOM

Questions
2.2 - It was comprehensive enough to support the activities defined in BPSOM
2.3 - The structure provided including the MINERVAdefs and MINERVAexample projects facilitated the work in the IDE
2.4 - The SoaML plug-in provides the support needed for the work in the IDE
2.5 - The iS4BPe plug-in was useful in making the BPMN2 model executable
<b>3 - About the QVT transformations included in Eclipse MINERVA design</b>
3.1 - Allows service models to be generated that suit your needs
3.2 - The correspondence between elements fir in with the ones you would model yourself manually from the BP model
3.3 - The value of automating this task for your organization is
<b>4 - About the implementation of services on the SoaML model and BPs in BPMN2</b>
4.1 - The manual implementation is facilitated by means of the SoaML service models
4.2 - The code generation from SoaML models by means of ModelPro and MagicDraw was useful
4.3 - The implementation of the BPMN2 model required more effort than expected
<b>5 - About the use of MINERVA (BPSOM + QVT transformations + Eclipse MINERVA design)</b>
5.1 - The prototype developed was useful for your organization
5.2 - It is feasible to use it in your organization as a basis for the service-oriented development from BPs
5.3 - Indicate the main characteristics of MINERVA that provide support to your organization objectives and your evaluation of them
5.4 - Provide any comments you have about MINERVA proposal

It was planned to collect the data after the case study was finished so that the participants were able to asses the complete development of the project.

### 10.3.6. Analysis

The analysis of the data was carried out by reading the answers in the questionnaire and consolidating them, although there were only two questionnaires to process, as they were answered by the leader of the project and the senior development who were the key people participating in the case study; this analysis is summarized below:

#### 1. About the guidelines provided for the modeling of BPs in the organization

For this question the answers were in the scale values of 4 and 5, indicating that the guidelines, references and notation BPMN2 were Quite and Much useful in carrying out the modeling of BPs in the organization. With respect to the tools used, they modeled the BP in two of the options we provided: Activiti Modeler (which allows the model to be saved in BPMN2 format) and Bizagi (which allows the model to be exported as image), the first one was evaluated as Quite useful, and the second one as Medium, as one of the main interests for this activity was the possibility of saving the BP model in BPMN2 format.

#### 2. About the Eclipse MINERVA distribution

For the usefulness and completeness of the tools support provided for the activities defined in BPSOM the answers were in the scale of 3 and 4, indicating that the provided support were Medium and Quite useful, and comprehensive. The structure provided including the .ecore models, the QVT transformations and the example project and was evaluated as 4 - Quite, indicating that it facilitates the work in the IDE, and the functionality provided by the Eclipse SoaML plug-in included in the distribution was also evaluated as 4 - Quite. The iS4BPe plug-in was not used to make the BPMN2 executable.



### 3. About the QVT transformations included in Eclipse MINERVA design

The suitability of the generated SoaML service models and the correspondence rules associated were evaluated as 4 and 5 in the scale, indicating that the QVT transformations provide the users with service models that fit to what they expect to design by themselves Quite and Much. The value of automating this task for the organization was indicated as 4 on the scale indicating that it is of high value.

### 4. About the implementation of services on the SoaML model and BPs in BPMN2

In this section only the first question regarding the facilitation of the implementation of services based on the generated service models was evaluated as 4 - Quite on the scale, as the automatic generation of code was not used. The effort to make the BPMN2 model executable was evaluated as 3 - Medium as this was also performed manually.

### 5. About the use of MINERVA (BPSOM + QVT transformations + Eclipse MINERVA design)

This section asked for a global evaluation of the use of MINERVA framework by means of BPSOM, the QVT transformations and the Eclipse MINERVA design distribution, whose feasibility to be applied in the organization as a basis for the service-oriented design from BPs was evaluated as 4 - Quite on the scale. The usefulness of the developed prototype was evaluated as 3 and 4 in the scale, indicating an average and quite high valuation of the development obtained. As for the open questions:

a - in the first place regarding the main characteristics of MINERVA that provide support for the organization objectives, these were indicated as:

a.1 - Traceability from the BPs to the services implementing them - valued as 5 - Much, the high value.

a.2 - Generation of documentation for a SOA - valued as 5 - Much, the high value.

a.3 - Automation of the services modeling - valued as 4 - Quite, a high value.

a.4 - Implementation of modeled services - valued as 3 - Medium, average valuation.

b- the feasibility of using MINERVA to support the automatic generation of SoaML service models from BPMN2 models in the organization was stressed, also some difficulties on the integration of some steps were mentioned, for example between the XML/XMI files of the BPMN2 model to be loaded in the IDE, but the technical guides provided were also mentioned as useful for overcoming these difficulties. In addition, the presence of a consultant with complete knowledge of BPSOM and the tools to be used is needed, to coach the development team and help them to overcome the difficulties.

In Figure 10.11 the maximum and minimum for the answers given are provided in diagram form, just to illustrate the complete evaluation, taking into account that the questions involving the specification of tools or characteristics are not included. As can be seen the general evaluation is on the higher part of the defined scale, equal to or above the 3 - Medium.

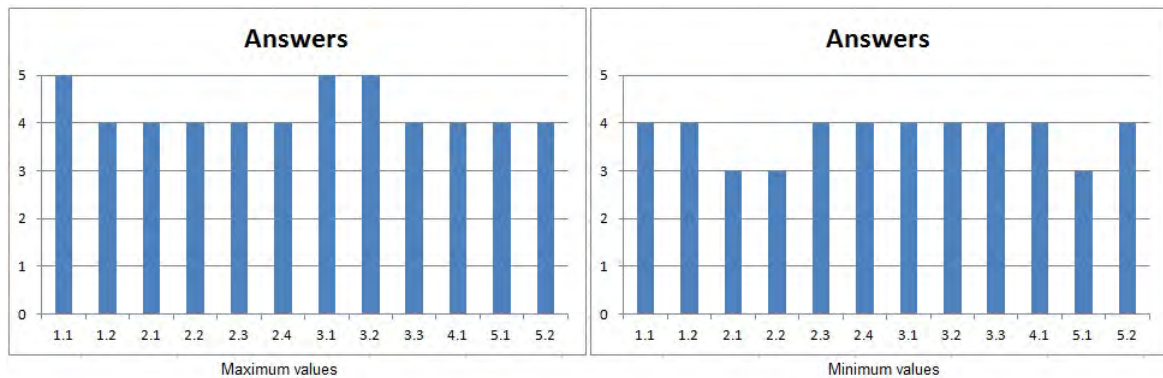


Figure 10.11.: Maximums and minimums of answers given in the questionnaire

As final considerations on the case study the participants stressed that an important advantage of BPSOM was that from each of the different phases in the development process documentation is generated, which in itself is an advantage, but it is also advantageous that each phase can be carried out easily by different teams or roles, as every activity generates an output which is used as input for the next one. This allows there to be a clear separation of tasks which fits the way and processes of work they had in the organization, so it is feasible for it to be applied in the entire organization.

### 10.3.7. Validity

The validity threats that could affect the case study were analyzed as part of the planning and are described below:

- Construct validity:
  - The questions included in the questionnaire to assess the case study were based on the research questions defined, and to avoid misunderstandings in the interpretation of the answers a scale was defined for them, apart from open questions to be answered freely by the participants. The topics to which the questions refer were defined based on desirable characteristics to be provided by a framework like MINERVA, such as guidelines, modeling notations, automation of tasks and tool support.
- External validity
  - The organization in which the case study was carried out presents several characteristics of organizations that would be interested in applying MINERVA, such as an IT area with several development projects and teams, with different infrastructures to support development efforts.
  - The fact that only two people answered the questionnaires prevent us from generalizing the results presented until another case study could be carried out to confirm the trends perceived.
- Reliability
  - This threat is concerned with to what extent the data and the analysis are dependent on the specific researcher. Threats of validity of this type could occur if it is not clear how to code collected data or if questionnaires or interview questions are unclear, which we took into account when defining the questions and the scale for the answers.

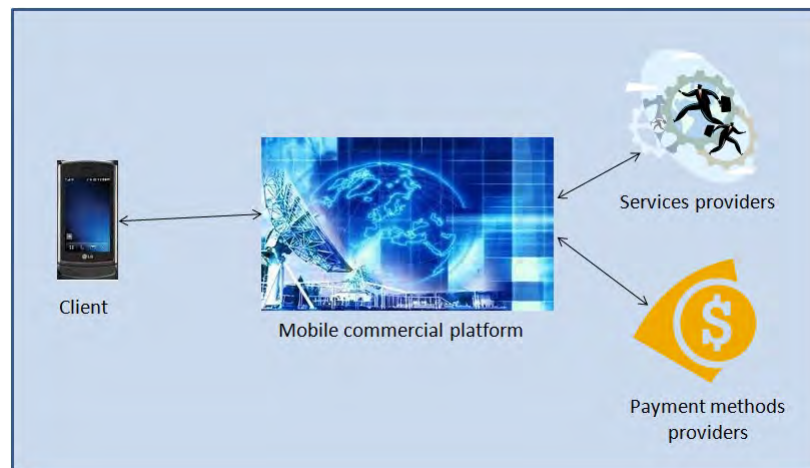
### 10.3.8. BPSOM use in the case study

In this section the use of MINERVA by means of BPSOM and its defined elements is described as presented in chapter 7, describing the activities performed, the roles and the artifacts generated, as well as presenting some of these. As the specific BP model is confidential we present a general textual description of it, along with some of the generated SoaML service models to realize it.

#### 10.3.8.1. Selling products on a mobile commercial platform

In this process the following participants are involved: the Client, the mobile commercial platform, the payment methods and the services providers. The organization corresponds to the mobile commercial platform which interacts with the other participants, and is organized internally in several software modules. Before being able to use the mobile as electronic wallet, the Client registers him/herself in the platform selecting a payment method offered by the platform which will be used to charge the purchases. The interaction with the Client is via SMS or a specific mobile application with menus, in both cases the format of the messages is pre-established so as to be able to recognize the product that is being purchased.

The process begins when a Client sends an SMS requesting the purchase of a product, the message is validated by the platform and if the checking of the client's data is successful the payment/s method/s available for him/her are shown to be selected by the Client. After the Client selects the payment method the platform asks for the PIN number associated to the Client, which is a security number provided to the Client when the registration was carried out. Then the platform first interacts with the payment methods providers to charge the Client with the corresponding amount, and after that with the services provider to request the product in question. If both interactions are successful the Client is charged with the corresponding amount and the requested product is delivered. The unsuccessful cases are not mentioned here but were modeled in the BPMN2 model, whenever a check was performed on the Client's data, payment method, balance and products. In Figure 10.12 the general view of the BP is presented, showing the interactions between participants.



**Figure 10.12.:** General view of the case study BP based on participants interactions

### 10.3.8.2. BPSOM application and tool support

Below the execution of BPSOM activities is described as presented in chapter 7. In the first place the Business Modeling activities had to be carried out, in the case study the BM1 - Asses the target organization was not performed as the project team had enough knowledge of the organization, roles, infrastructure and other information required. The BM2 - Identify business processes activity was carried out by the Business Analyst/Analyst and Architect from the software team, to model the selected BP in BPMN2 using the Activiti Modeler tool and saving the model in BPMN2 format.

After this, the Design activities were carried out, by applying the model-driven approach defined. In D1- Identify and categorize services the activities to be realized by services were marked with the "ServiceTask" type by the Architect, and the model was loaded in the Eclipse MINERVA design. Following the technical guides provided, the QVT transformations were executed generating the SoaML service models corresponding to the BPMN2 model, with the diagram corresponding to this activity being the one for ServiceArchitecture with the definition of participants, services and roles.

The SoaML plug-in was then used to import the XMI file obtained from the transformations and visualize the folder structure and the corresponding SoaML diagrams generated. In Figure 10.13 the ServicesArchitecture generated is shown. As the automatic generation of services was applied, the activity D2 - Specify services was already performed since the service model generated included the diagrams for the complete specification of services (interfaces, operations, parameters, messages and contracts). In Figure 10.14 some of the Service Contracts generated are shown and in Figure 10.15 some of the generated Service Interfaces can be seen.

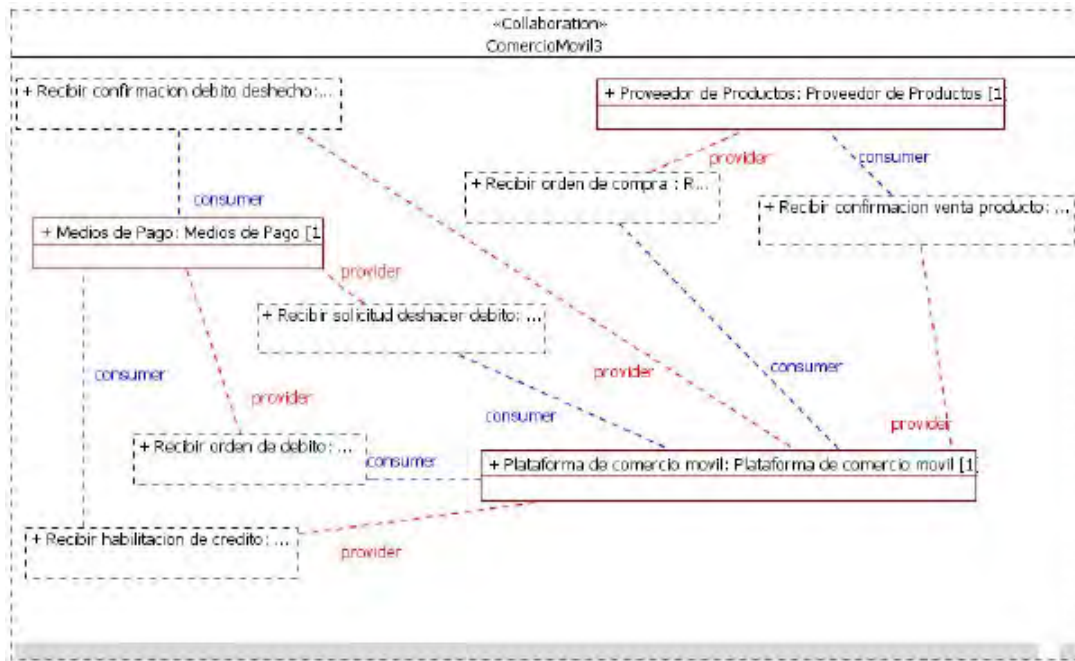


Figure 10.13.: Generated SoaML Service Architecture diagram

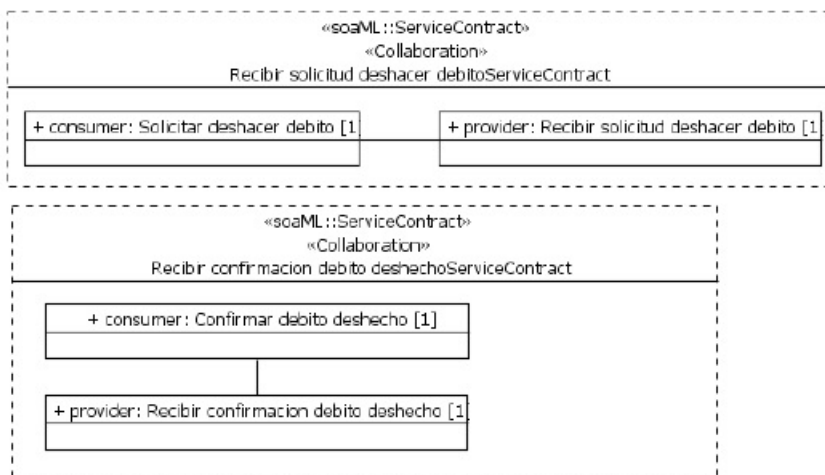


Figure 10.14.: Generated (some) Service Contracts and Interfaces

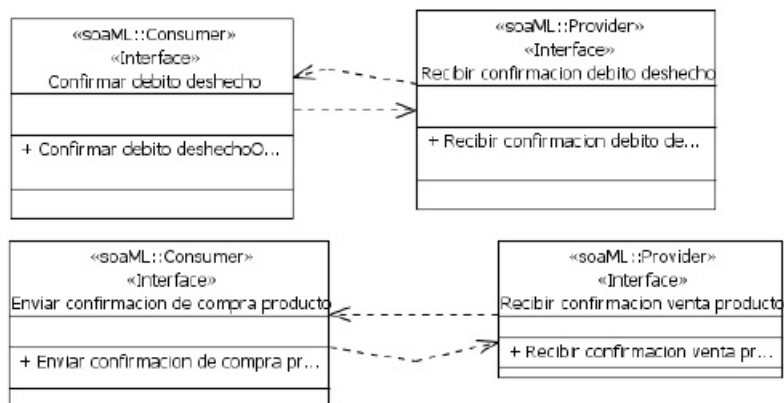


Figure 10.15.: Generated (some) Service Interfaces diagrams

The activity D3 - Investigate existing services was not performed as the development of services is somewhat new in the organization and this BP in particular has no existing services to be reused. The activity D4 - Assign components to services was to be carried out using the Eclipse Development distribution with ModelPro and MagicDraw to define the deployment nodes and components, but there were some difficulties in the integration of the two tools so they decided to carry out the implementation of services manually.

The activity D5 - Define interactions of services was not performed as the team decided they did not need the vision of services interaction since they had the BPMN2 model to assess the interaction based on the activities modeled. For the implementation of the BPMN2 model and the services generated they decided to use the development process of the organization, which they already knew and had already used many times, so in the end, the implementation was not carried out for the complete BP, as their main interest was in the Business Modeling and Design activities, artifacts and tools support.

### 10.3.9. Conclusions and lessons learned

Based on the presented analysis of the questionnaires we can answer the research question concluding that the MINERVA proposal for carrying out service-oriented software developments from BPs in organizations, using BPSOM and automatic generation of SoaML service models from BPMN2 models, is useful for organizations such as the one involved in the case study. BPSOM was evaluated as providing a better way to undertake this kind of development projects for the Business Modeling and services Design activities, which were the ones applied. Nevertheless, the interaction of some inputs/outputs needs further work so manual intervention is minimized.

The QVT transformations defined to generate SoaML service models from BPMN2 models were also evaluated as appropriate and useful to the software designers in the development process, and the tools support provided was also appropriate and useful in carrying out the activities in BPSOM. Nevertheless, we are not able to generalize the results as the case study consists of a pilot project on the development of a prototype for just one BP in the organization, and by only one development team. Collaboration to carry out another case study in the organization next year is to be proposed, in several development projects, if possible.

As lessons learned this case study showed us that people need to be trained in the notations, in SoaML in particular, as it is the one less well-known by software modelers, and although it is a UML profile it is not that easy to understand even for people who have knowledge of UML. BPMN2 was not difficult to understand and they were able to model BPs following best practices and taking into account the workflow patterns presented, using the tools selected.

Defining the Eclipse MINERVA design distribution was appropriate for the support of the activities, and they were able to use it by themselves following the technical guidelines, although the generation of code integrating the MDA engine could not be performed, as the tools present some difficulties with the integration and the versions of the Eclipse IDE. The manual development of services can nevertheless be performed from the SoaML service models, which was valuable for the project.

The chain of development where the output from one activity is the input to the next one by means of the generation of artifacts based on standards and applying standard formats and transformations is also valuable for the project and helps improve the productivity of the realization of the defined activities, such as not having to specify the SoaML service models manually.

## 10.4. Empirical validation of BPCIP

To assess the applicability of BPCIP focusing on the execution measures from BPEMM and the execution measurement and improvement activities defined, a case study was carried out in a real organization, corresponding to the definitions of Part 1.a and Part 2 presented in section 10.1. Our

aim was to see whether the provided guides, execution measures and the set of tools to support the activities defined are appropriate and useful for a real organization from the point of view of the business people.

The organization in which the case study was carried out is the Hospital General de Ciudad Real<sup>7</sup> (HGCR), which is a public hospital that integrates the network of Servicios de Salud de Castilla - La Mancha<sup>8</sup>, (SESCAM, Health Services of Castilla - La Mancha). The direct influence area of the HGCR consists of 42 municipalities from the Ciudad Real province with approximately 300.000 inhabitants in all, but some services are provided to the complete autonomous region community which has more than 2.000.000 inhabitants. It has around 2.600 employees including MDs, nurses, laboratory technicians, administrative staff and several other categories. The HGCR started a project with the UCLM in 2007 which included the modeling of several BPs with BPMN, the definition and evaluation of characteristics to select a BPMS, and the implementation and deployment of the BPs modeled.

The case study was carried out in the context of this project with the Quality division of the HGCR, for which a representative BP process was selected from the ones already modeled by a previous work. In the following the definition, execution, data collection and analysis of the case study is presented based on the guides in [Yin, 2002, Brereton et al., 2008].

### 10.4.1. Background

The research on the topic of BPM, continuous improvement and execution measurement was carried out at the beginning of this thesis, using the bibliography studied which had been selected by experts, as presented in chapter 3.

The main research question for this case study is stated as:

*Does MINERVA provide, by means of BPCIP and BPEMM, a useful proposal for carrying out BP continuous improvement based on execution measurement of BPs in organizations ?*

Additional research questions derived from this are defined as:

- are the execution measurement and improvement activities proposed in BPCIP appropriate and useful for business people in the management and improvement of BPs in the organization?
- are the execution measures integrated in BPEMM appropriate and useful for business people to be able to obtain information on the execution of the BP?
- is the tools support provided to implement, execute and analyze the execution measurement results appropriate and useful for the software team and for business people respectively?

### 10.4.2. Design

The type of case study carried out was a single-case in a single organization and in a single project of the organization, corresponding to a holistic case [Yin, 2002]. The object of the study was the execution measurement, analysis and improvement of a BP in the context of the Hospital and the pre-production environment (laboratory) defined for this thesis as described in Part1.b of the case study in section 10.1, with the unit of analysis being the quality group of the HGCR. Several other sub-questions were defined based on the additional research questions for the specific organization and project:

- are the execution measures provided in BPEMM appropriate for the organization and for the indicators already defined to asses the execution of BPs in the organization ?

---

<sup>7</sup><http://www.hgucr.es/>

<sup>8</sup><http://sescam.jccm.es>

- is the tool support provided by means of the three process engines selected (Activiti, Bonita, Intalio) for each of the languages for execution (BPMN2, XPD, BPEL) feasible and do they provide:
  - all support needed for implementing and executing the BP in the language provided?
  - support for registering execution data from the BP cases and extracting it to be able to perform the analysis?
  - interoperability with the ProM tool to be able to load execution data into the framework for further analysis ?
- is the tool support provided by means of the ProM BPEMM plug-in feasible and could it be useful for carrying out the analysis of BPs execution and does it provide the needed functionality?

The data collected to answer these and the research questions defined previously was qualitative, obtained by means of interviews with the Responsible of the quality group of the HGCR who is also the M.D. expert Responsible of the selected BP, and it is also obtained by the proofs carried out using the pre-production environment (laboratory) defined for implementation and execution of BPs.

### 10.4.3. Case selection

The Patient Major Ambulatory Surgery (MAS) from the organization was selected for several reasons: first of all it involves several sections in the organization and has been modeled in BPMN in a previous stage of the project so the existing model could be used; secondly there are several indicators defined for the BP so they can be compared with the execution measures integrated in BPEMM; and thirdly, the quality group and the responsible of the BP is interested in implementing this BP to be executed in a BP engine. As the organization is still in the process of selecting a BPMS, it is of interest to develop a proof of concept in a pre-production environment (laboratory), and evaluate the functionality for registering execution data, to extract it and load it into the ProM tool.

### 10.4.4. Procedures and roles

The main roles applied in the case study were the Responsible of the BP from the quality group of the HGCR, and the Responsible of the Improvement carried out by this author. The Patient MAS BP model was given by the quality group and it was adapted to perform the implementation in the pre-production environment for the three BP engines selected: Activiti, Bonita and Intalio, and to run the simulation in the CPNTools enabling the generation of an adequate amount of execution logs to be analyzed. The indicators already defined for the Patient MAS BP were also given to be compared to the ones integrated in BPEMM for evaluation by the quality group.

The given Patient MAS BP was specified in BPMN in Bizagi and exported into XPD format, although it could not be loaded into the process engine implementing this language, Bonita. It was also transformed into BPEL to be loaded into the process engine implementing this language, Intalio, but it could not be loaded either. The same occurred with the transformation to BPMN2 so the Patient MAS BP was modeled directly in the designers the process engines provide.

Several executions of the BP were performed in each process engine in the laboratory, registering the corresponding execution data, which was extracted in .csv format and transformed into MXML format by means of the ProMImport framework and the Fluxicon Nitro tool, to assess the feasibility of being loaded in the ProM framework for further analysis. As mentioned in section 10.1 these BP executions were not real and too few, so a simulation of the BP in CPNTools was carried out. For this, the information on the parameters needed such as activities duration and performers and number of Patient MAS performed by day was obtained from the quality group of the HGCR.

CPNTools allows the execution logs in MXML format to be registered, one by each BP case, and by means of the ProMImport framework these execution logs were merged into one to be loaded into the ProM framework for further analysis. To do this, among other functionalities of ProM, the prototype of the ProM BPEMM plug-in was used, to be able to find improvement opportunities. These were integrated into the BP generating a new version of the BP which was simulated once more to be compared and to assess whether the improvements integrated allow the desired goals to be achieved.

This researcher acted as a consultant on MINERVA leading the implementation of the BP in the pre-production environment (laboratory) and running the simulation of the BP in CPNTools for the two versions of the BP. The characteristics of the BP engines were assessed directly by the groups carrying out their evaluation and presented as a report of the results of their work. After both the evaluation and the simulation were finished, an interview was carried out with the Responsible of the quality group and Responsible of the BP presenting the results for him to evaluate the perceived utility of the proposal.

#### **10.4.5. Data collection**

The collection of data was performed from the report generated for the execution of the Patient MAS BP in the BP engines carried out by the groups in the pre-operation environment (laboratory) as well as from the answers in the interview with the Responsible of the quality group and Responsible of the BP. The data of the case study carried out in the BP engines was registered during the execution and is available in the corresponding monograph (in Spanish only), and the answers from the interview were registered during the interview. In the interview the results of the simulation of the BP were shown to the responsible along with the execution measures applied, as well as the suggestion for the improvement of the BP and the new results obtained.

#### **10.4.6. Analysis**

The analysis of the data was carried out from the reports of the BP engines evaluation and the answers in the interview. The Patient MAS BP from the HGCR was implemented and executed in the BP engines selected: Activiti, Bonita and Intalio, registering the process and evaluation of the execution, which is summarized below.

For the Activiti process engine, the following characteristics can be highlighted from the execution of the case study: all components are web-type providing several advantages such as independence from the platform, clear and easy to use interfaces for each component, clear error messages, and clear functionality assigned to each component making the process of implementing and executing the BP easier. Some elements from BPMN2 are not supported yet, such as definition of pools and consequently, message flows between activities; many things in user forms have to be implemented directly in the BPMN2 XML file and the mechanisms for monitoring and managing the execution of BP cases is insufficient as it does not provide much information. Data from the execution was easily extracted to be loaded in the ProM framework.

For the Bonita process engine, the following characteristics can be highlighted from the execution of the case study: its versatility for modeling processes that may have a significant complexity in its business rules, strength in the execution of the processes modeled in terms of compliance with all the steps defined and following the defined flow correctly, ability to interact with other products through the facility provided by connectors, ease of monitoring progress and status of processes, structure of roles which is interesting in defining user profiles. Some difficulties were detected for the implementation of user forms, the execution of activities based on people assigned to roles and the fact that some constructions of BPMN2 are not available (although it executed XPDLL the modeler provides a subset of BPMN2). Data from the execution was easily extracted to be loaded in the ProM framework.

For the Intalio process engine, the following characteristics can be highlighted from the execution of the case study: it presents a stable behavior and a friendly environment, workspaces for each



user interface presents a simple and intuitive task assignment for the execution of workflows, while the design of the process in the designer presents some features like the BPMN notation certain modifications have to be performed in order to include web-services forms and processes, the server administrator can view a log of all events involved in the execution of a request including the timestamp, activation events, beginning and end of each task, but user information does not appear in the event so we can not know who executed the job, which is an important drawback of the engine.

In the interview with the Responsible of the quality group and Responsible of the BP, the execution measures defined were discussed and were evaluated positively, as they include and extend the indicators they already had for the BP. As an example, one of the main indicators they are interested in is the occupation of the OR, which is covered by the execution measure M9 (derived) for the Capacity utilization for a resource in the time dimension of the BP Generic view, as presented in chapter 6. The complete lifecycle of BPCIP was discussed and the simulation results for both versions were presented in the ProM plug-in, to show how the complete cycle would be (although the ProM plug-in is a prototype and it only shows the measures for Throughput Time (TT) of the BP from the time dimension of the Generic view by now). It was positively perceived and feasible for integration in the organization, which we would asses in a future case study.

As for the evaluation of the ProM plug-in carried out in the laboratory by this author, what we wanted to asses was to the feasibility of implementing the BPEMM measures as a ProM plug-in, and to be able to analyze the results in the three dimensions defined by the BPEMM cube as presented in chapter 6 which was all positively achieved.

#### 10.4.7. Validity

The validity threats that could affect the case study were analyzed as part of the planning, which are similar to the ones presented in section 10.3, and are described below:

- Construct validity:
  - The questions asked in the interview to asses the case study were based on the research questions defined, and the topics to which the questions refer were defined on the basis of desirable characteristics to be provided by a framework as MINERVA, such as guidelines, execution measurement and improvement activities as well as tools support.
  - The characteristics assessed for the execution of the BP selected in the process engines selected were defined on the basis of desirable characteristics to be provided by these tools as defined in many existing evaluations and guides, as presented in chapter 9.
- External validity
  - The organization in which the case study was carried out presents several characteristics of organizations that would be interested in applying MINERVA, such as several BPs to be executed in a BPMS from which to gather execution data for analysis to find improvement opportunities. It already has a quality group which is a key factor for improvement efforts.
  - Although the interview was with only one person, which is the Responsible of the quality group and of the BP selected, his answers can be considered as expert opinion on the subject, as he leads quality and improvement efforts in the organization. Nevertheless we are aware that the opinion of only one person does not allow us to affirm anything but that the BPCIP proposal could be useful in such an organization.
  - The fact that the IT area could not participate in the implementation of the prototype of the selected BP and that we had to perform a simulation instead is also a threat, so when the IT area can participate it should be actually implemented and executed within the organization.
- Reliability
  - the same as for the previous BPSOM case study presented in section 10.3.

### 10.4.8. BPCIP use in the case study

In this section the use of MINERVA by means of BPCIP and its defined elements is described as presented in chapter 5 including activities performed, roles and artifacts generated, presenting some of these. The original Patient MAS modeled previously with the HGCR was implemented in the three process engines selected as-is, modeled in each process engine designer. But for the simulation it has to be adapted, as the original presented many manual activities so we add some activities to be realized by services.

#### 10.4.8.1. Patient MAS in HGCR

This BP involves the participants Patient, MAS Unit which in turn is made up of the participants Secretary, Nurse and Auxiliary Nurse, and the Surgical Block which in turn is composed of the participants Surgeon, Anesthetist, instrument Nurse, Operation room (OR) Attendant and Operation room (OR) Auxiliary. The first sub-process in the HGCR is the “Admission and Registration” in which the patient goes to the hospital to have the surgery, secondly the “Preparation for MAS” in which the patient is given the clothes and the particular place in the list for surgery, after this the “Pre-intervention” and the “Intervention” (the surgery) are performed, then the “Post-intervention”, the “Observation and recovery” and finally the “Release Patient”. In Figure 10.16 a global view of the Patient MAS as modeled by the HGCR staff is presented, which is available on line in<sup>9</sup> including all the defined sub-processes (in Spanish only).

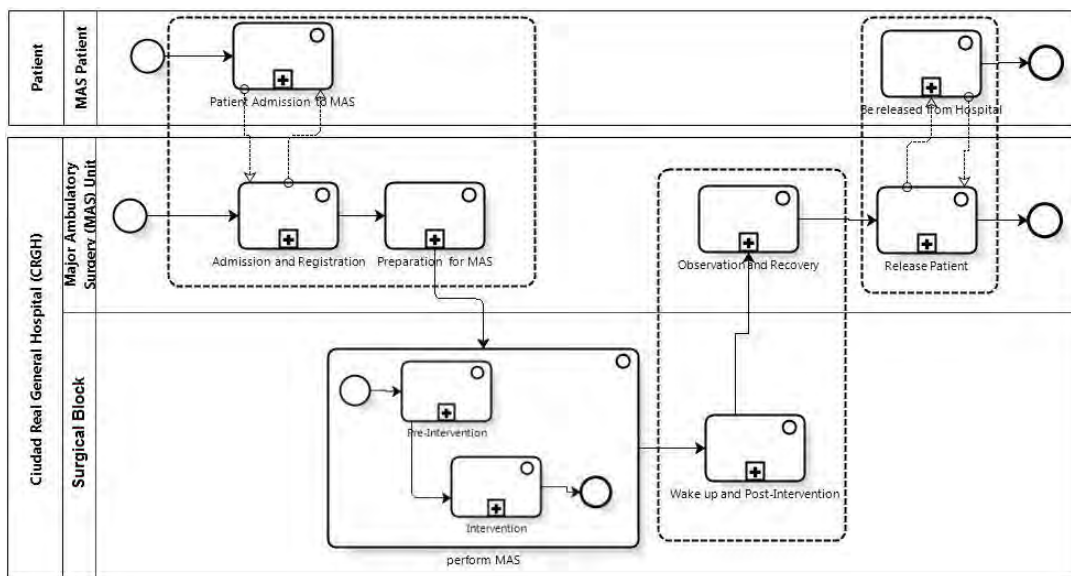


Figure 10.16.: Complete Patient MAS BP from HGCR

#### 10.4.8.2. BPCIP application and tool support

The execution of BPCIP activities is described below as presented in chapter 5. As the model is already specified the activities from the Business Modeling Discipline in the Design&Analysis phase are not executed as defined, but only to specify the selected sub-processes from the BP model into each of the process engines, and the adapted one as Petri Net for the simulation.

The activity EM1 - Select execution measures was evaluated in the meeting with the Responsible of the quality group of the HGCR based on the discussion of the indicators they already have and the execution measures proposed. As the ProM plug-in only implements the Throughput Time (TT) measure for now, these were the only ones we could evaluate. The BP execution Average TT goal was defined to be under 90 minutes, the Warning rank between 90 and 120 minutes and the

<sup>9</sup><http://161.67.140.34:82/cma/>

Problems rank to be above 120 minutes, for the sub-processes “Admission and Registration” and “Preparation for MAS”, which were the ones modeled for execution and simulation.

The Configuration phase corresponds to the implementation carried out in the pre-production environment, for which the first two sub-processes of “Admission and Registration” and “Preparation for MAS” were modeled for Activiti and Bonita, and the “Pre-intervention” sub-process was modeled in Intalio, as the main objective for the implementation was to assess the feasibility of executing the Patient MAS BP in different process engines and languages, registering the execution and extracting the execution data to be loaded into the ProM framework. The implementation in Activiti is presented as an example, the ones for Bonita and Intalio can be seen in Appendix E. Although the complete Patient MAS presented in Figure 10.16 is in English, the case study was carried out in Spanish, so the figures presented are in this language as taken from the implementation. The activity I1 - Implement BP with services was performed but mainly for making the BP model executable, as BPSOM was not used here.

The Execution phase is divided in two parts: in the first one the executable BP model as implemented previously for each BP engine was executed and the execution data for each BP instance was registered, as defined by the EM2 - Implement execution measures collection and EM3 - Collect execution measures. In the second one, the BP model was specified as a Petri Net in the CPNTools and the information for the parameters such as duration of activities, resources etc. as defined by business people was added to it, and the simulation was carried out corresponding also to the BP execution and the measurement activities.

### Implementation and execution in Activiti

The Patient Admission and Registration and Preparation for MAS sub-processes from the Patient MAS BP were modeled in the Activiti Modeler tool and uploaded in the Activiti process engine to be executed, shown in Figure 10.18.

As Activiti does not allow several pools to be modeled in a BP, it was modeled in one pool with several lanes, to be able to execute it. The participants are the Patient, Secretary, Nurse and Auxiliary Nurse, whom carry out the activities for registering the Patient, giving out the surgery clothes and assigning the place for the surgery, also preparing the patient for the surgery performing tasks such as shaving the surgical site.

Most of the activities are of User type which means that they are executed by a person which has the corresponding role assigned. Based on the data defined to be gathered in each corresponding activity, several user forms were defined for the assigned user to fill. In Figure 10.17 an example of two user forms produced is shown with fields and information.

The figure shows two side-by-side user forms. The left form, titled 'Registro de Paciente', contains input fields for 'Nombre', 'Apellido', 'Fecha de Nacimiento', 'Sexo' (with a dropdown menu set to 'Masculino'), and 'Cedula'. It has 'Ok' and 'Cancel' buttons at the bottom. The right form, titled 'Informacion de procedimiento', displays pre-filled information: 'Nombre: Pablo', 'Apellido: Perez', 'Fecha Nacimiento: 1983-05-26', and 'Cedula: 1234567-8'. It has two large text areas labeled 'Detalle de Procedimiento:' and 'Detalle de Vestimenta:'. It also has 'Ok' and 'Cancel' buttons at the bottom.

Figure 10.17.: Example of User forms defined in Activiti

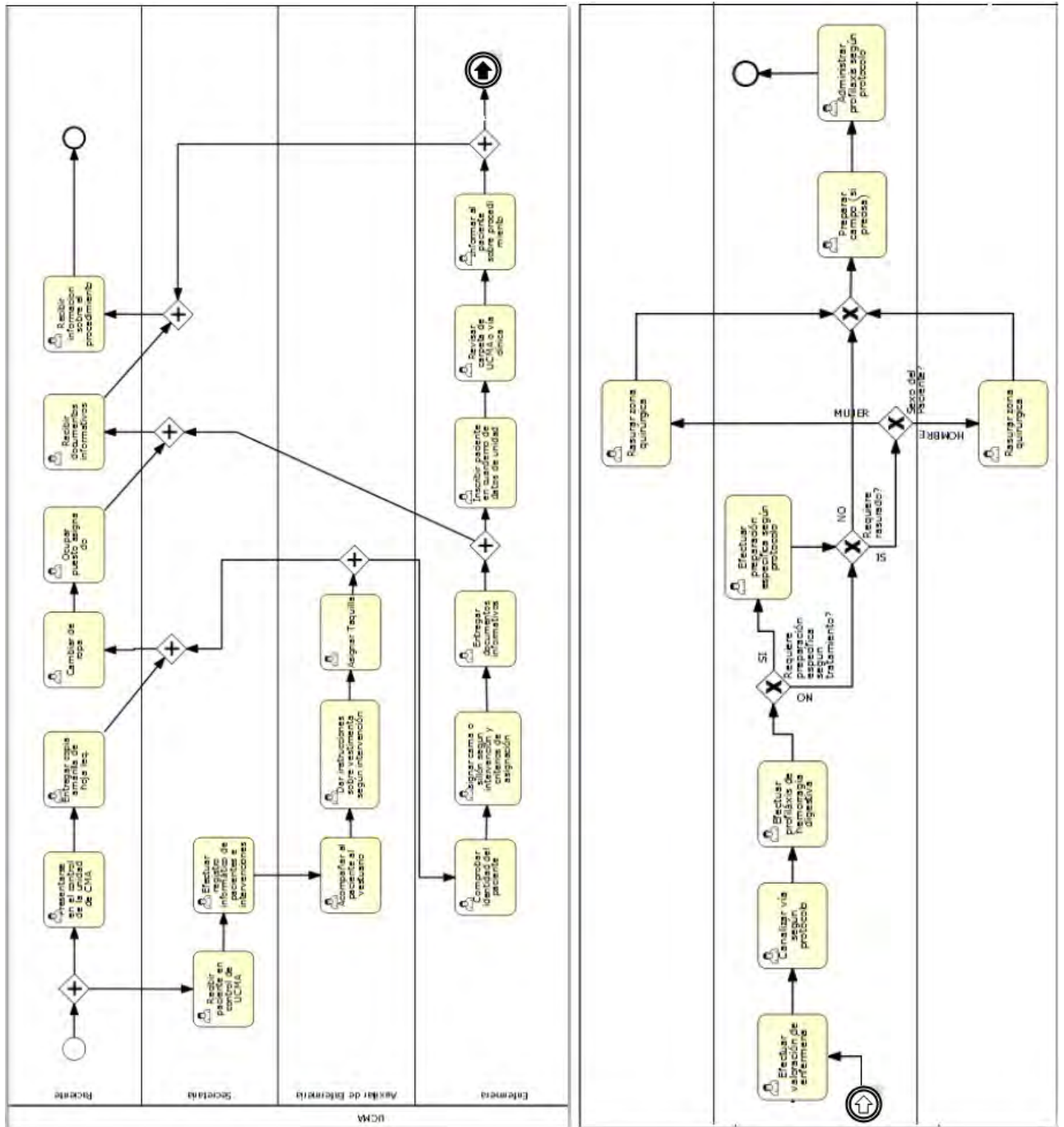
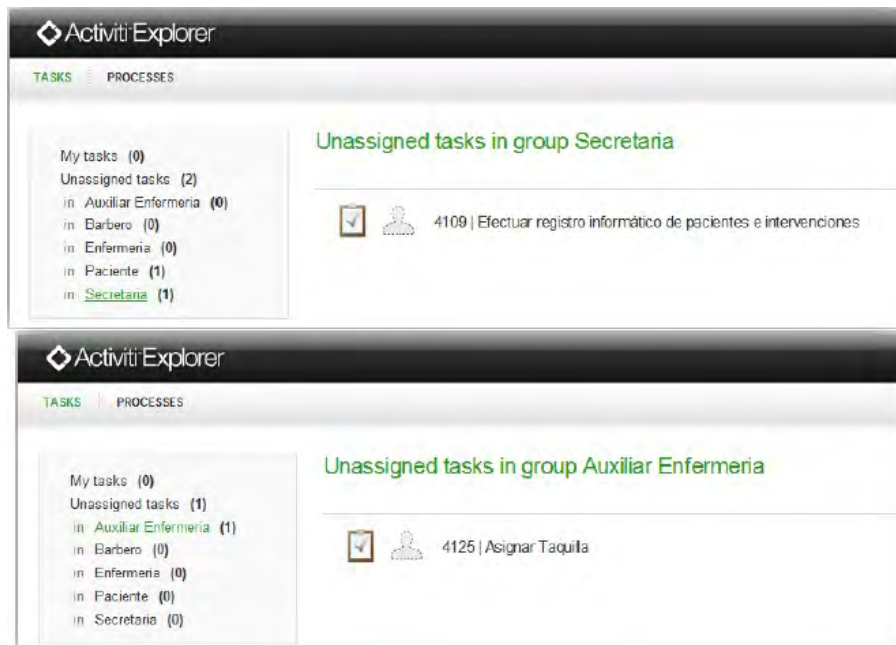


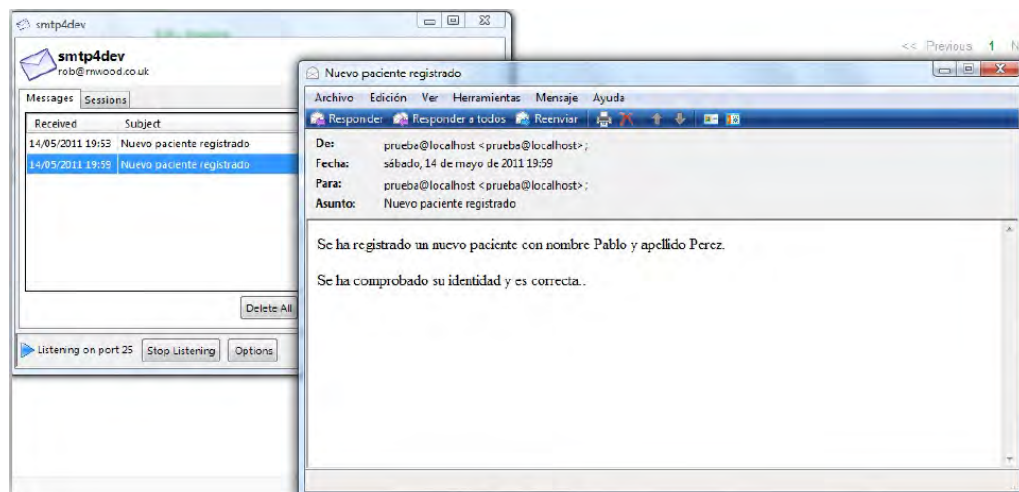
Figure 10.18.: Patient MAS sub-processes modeled in Activiti

Each participant can view the list of tasks she or he has assigned to perform as the process flow progresses, assigned to the corresponding role, and which have to be claimed by the specific person who will perform it. In Figure 10.19 an example of task lists is shown, on the top there are the unassigned tasks in the Secretary group, and below in the Auxiliary Nurse group.



**Figure 10.19.:** Example of task lists assigned to roles in Activiti

To test the implementation of ServiceTasks an activity was defined to send a mail which in Activiti corresponds to it, simulating the checking of the Patient identity. For this a mail server was installed and the configuration files needed in Activiti were set so that the process engine is able to invoke it. In Figure 10.20 the definition and sending of the mail is shown, on the left the mail server showing the emails received and on the right the corresponding data from one of the mails.



**Figure 10.20.:** Example of ServiceTask implementation as sending mail in Activiti

The registered data about the BP execution was gathered from the data base of the Activiti process engine, by means of queries on the corresponding tables to obtain the .csv file with the data. Activiti works with several existing data bases such as MySql or Postgress. For this case study the MySql data base was used and in Figure 10.21 one of the queries is shown.

```

CREATE ALGORITHM=UNDEFINED DEFINER='root'@'localhost'
SQL SECURITY DEFINER VIEW `activiti`.`instanciasactividades` AS
select `activiti`.`act_hi_taskinst`.`ID` AS `Id`,
`activiti`.`act_hi_taskinst`.`PROC_INST_ID` AS `InstanciaProceso`,
`activiti`.`act_hi_taskinst`.`NAME` AS `NombreActividad`,
`activiti`.`act_hi_taskinst`.`ASSIGNEE` AS `Usuario`,
`activiti`.`act_hi_taskinst`.`START_TIME` AS `FechaComienzo`,
`activiti`.`act_hi_taskinst`.`END_TIME` AS `FechaFin`,
`activiti`.`act_hi_taskinst`.`DURATION` AS `Duracion`
from `activiti`.`act_hi_taskinst`$$

```

Figure 10.21.: Example queries to obtain the execution data from Activiti

As the University of the Republic and the University of Castilla - La Mancha are partners in the Academic Initiative of Fluxicon, we were able to use this tool to transform the .csv file into the MXML format. In Figure 10.22 a screenshot of this transformation is presented, showing the loaded data obtained from the query.

ID	InstanciaProceso	NombreActividad	Usuario	FechaComienzo	FechaFin
1	3014	3010	Paciente1	12/05/2011 21:04	12/05/2011 21:04
2	3017	3010	Secretaria1	12/05/2011 21:04	12/05/2011 21:04
3	3023	3019	Paciente1	12/05/2011 21:04	12/05/2011 21:04
4	3026	3019	Secretaria1	12/05/2011 21:04	12/05/2011 21:04
5	3032	3028	Paciente1	12/05/2011 21:05	12/05/2011 21:05
6	3035	3028	Secretaria1	12/05/2011 21:05	12/05/2011 21:05
7	3041	3037	Paciente1	12/05/2011 21:05	12/05/2011 21:05
8	3044	3037	Secretaria1	12/05/2011 21:05	12/05/2011 21:05
9	3050	3046	Paciente1	12/05/2011 21:06	12/05/2011 21:06
10	3053	3046	Secretaria1	12/05/2011 21:06	12/05/2011 21:06
11	3059	3055	Paciente1	12/05/2011 21:06	12/05/2011 21:06
12	3062	3055	Secretaria1	12/05/2011 21:06	12/05/2011 21:06
13	3085	3010	Paciente1	12/05/2011 21:10	12/05/2011 21:10
14	3088	3028	Secretaria1	12/05/2011 21:10	12/05/2011 21:10
15	3076	3028	AuxiliarEnfermeria1	12/05/2011 21:11	12/05/2011 21:11
16	3079	3019	Secretaria1	12/05/2011 21:11	12/05/2011 21:11
17	3082	3046	Paciente1	12/05/2011 21:11	12/05/2011 21:11
18	3085	3010	Secretaria1	12/05/2011 21:13	12/05/2011 21:13
19	3093	3019	AuxiliarEnfermeria1	12/05/2011 21:16	12/05/2011 21:16
20	3096	3046	Secretaria1	12/05/2011 21:20	12/05/2011 21:20
21	3099	3037	Paciente1	12/05/2011 21:20	12/05/2011 21:20
22	3102	3055	Paciente1	12/05/2011 21:34	12/05/2011 21:34
23	3110	3010	AuxiliarEnfermeria1	12/05/2011 21:35	12/05/2011 21:35

Figure 10.22.: Fluxicon transformation of Activiti .csv log into MXML

Once the event log is transformed into the MXML format it can be loaded in ProM for further analysis, which is shown in Figure 10.23.

Log Summary	Duration	Completion
3138,3028.Dar instrucciones sobre vestimenta segun intervenci3n	12/05/2011 22:05,61549+complete	1, 0.99%
3066,3028.Efectuar registro inform3tico de pacientes e intervenciones	Secretaria1,12/05/2011 21:10,12/05/2011 21:11,63575+complete	1, 0.99%
3381,3037.Revisar carpeta de UCMA o v3-a cl3-nica	Enfermeria1,12/05/2011 23:17,12/05/2011 23:30,601077+complete	1, 0.99%
3236,3028.Cambiar de ropa	Paciente1,12/05/2011 22:45,12/05/2011 22:50,271658+complete	1, 0.99%
3197,3010.Asignar Taquilla	AuxiliarEnfermeria1,12/05/2011 22:33,12/05/2011 22:36,150964+complete	1, 0.99%
3206,3010.Cambiar de ropa	Paciente1,12/05/2011 22:36,12/05/2011 22:36,45679+complete	1, 0.99%
3200,3046.Asignar cama o sill3n segun intervenci3n y criterios de asignaci3n	Enfermeria1,12/05/2011 22:35,12/05/2011 22:49,845073+complete	1, 0.99%
3096,3046.Efectuar registro inform3tico de pacientes e intervenciones	Secretaria1,12/05/2011 21:20,12/05/2011 22:03,2613540+complete	1, 0.99%
3032,3028.Presentarse en el control de la unidad de CMA	Paciente1,12/05/2011 21:05,12/05/2011 21:35,1840127+complete	1, 0.99%
3050,3046.Presentarse en el control de la unidad de CMA	Paciente1,12/05/2011 21:05,12/05/2011 21:05,1840127+complete	1, 0.99%

Figure 10.23.: Activiti MXML log loaded into ProM

As the objective of this part of the case study was only to assess the feasibility of obtaining the needed execution data from the process engine and to be able to load it into the ProM framework, to be analyzed with the ProM BPEMM plug-in, no further analysis in ProM is needed at this stage.

### Simulation with CPNTools

The simulation of the Patient MAS in CPNTools was performed to be able to obtain a considerable amount of execution data for the BP to be further analyzed by means of the ProM BPEMM plug-in prototype, so assessing its feasibility. If we had been able to execute the BP in the HGCR it would not have been necessary to simulate the BP, as we could have gathered real data from the execution to be analyzed by means of ProM. In the executions in the three process engines presented in the previous sections we have proven that it is possible to obtain the execution data from the engines and to load it into ProM.

The simulation in the CPNTools environment along with the extension for MXML logging defined by the ProM framework allow us to obtain the event logs in the MXML format to be loaded in the BPEMM ProM plug-in for the calculation and visualization of several of the selected measures. The BP was modeled as Petri Net and enhanced with data that we have collected previously to carry out the simulation from the real execution of the BP in the Hospital.

For the simulation an adapted version of the Patient MAS BP was modeled since most of the activities in the original one are manual, so we defined several service activities between pools. The adapted Patient MAS model is the one we have also used as a basis for the examples in chapter 6 and in chapter 8, it is thus presented in Figure 10.25 modeled in Oryx editor as we needed to save it in BPMN2 format.

To model the petri net we have followed the proposals in [Rozinat et al., 2008, van der Aalst et al., 2010] by defining a hierarchical petri net providing a global view of the collaboration between the participants in the BP, but logging only the execution corresponding to the Hospital and the realization of the BP by services. In Figure 10.24 the global view of the petri net corresponding to the Patient MAS is shown, with the participants in the collaborative BP and their interaction.

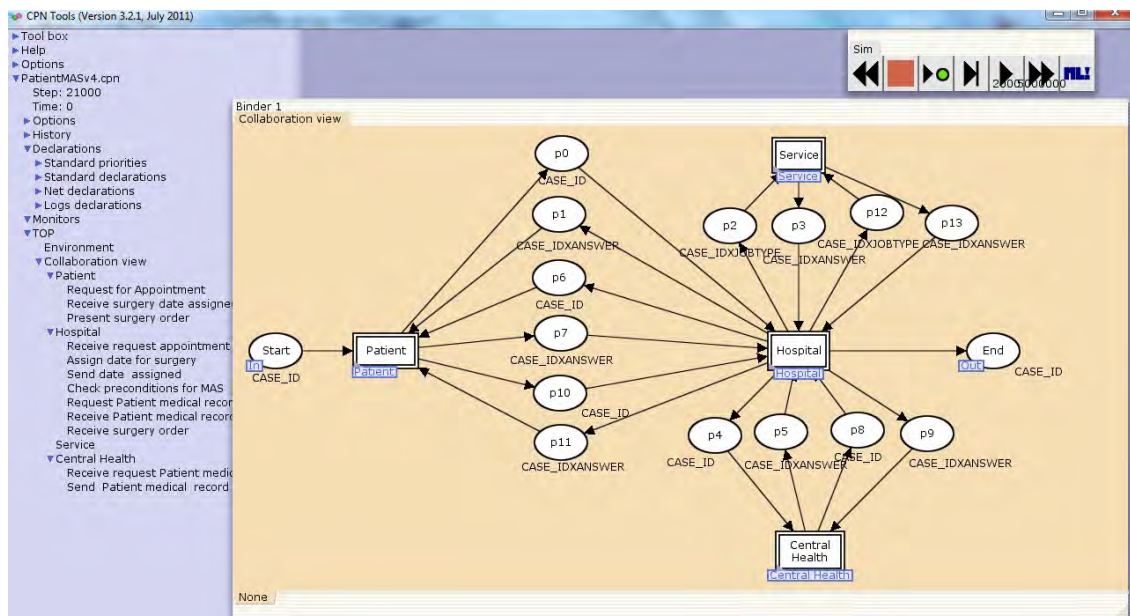


Figure 10.24.: Global view of the Patient MAS Petri Net defined in CPNTools

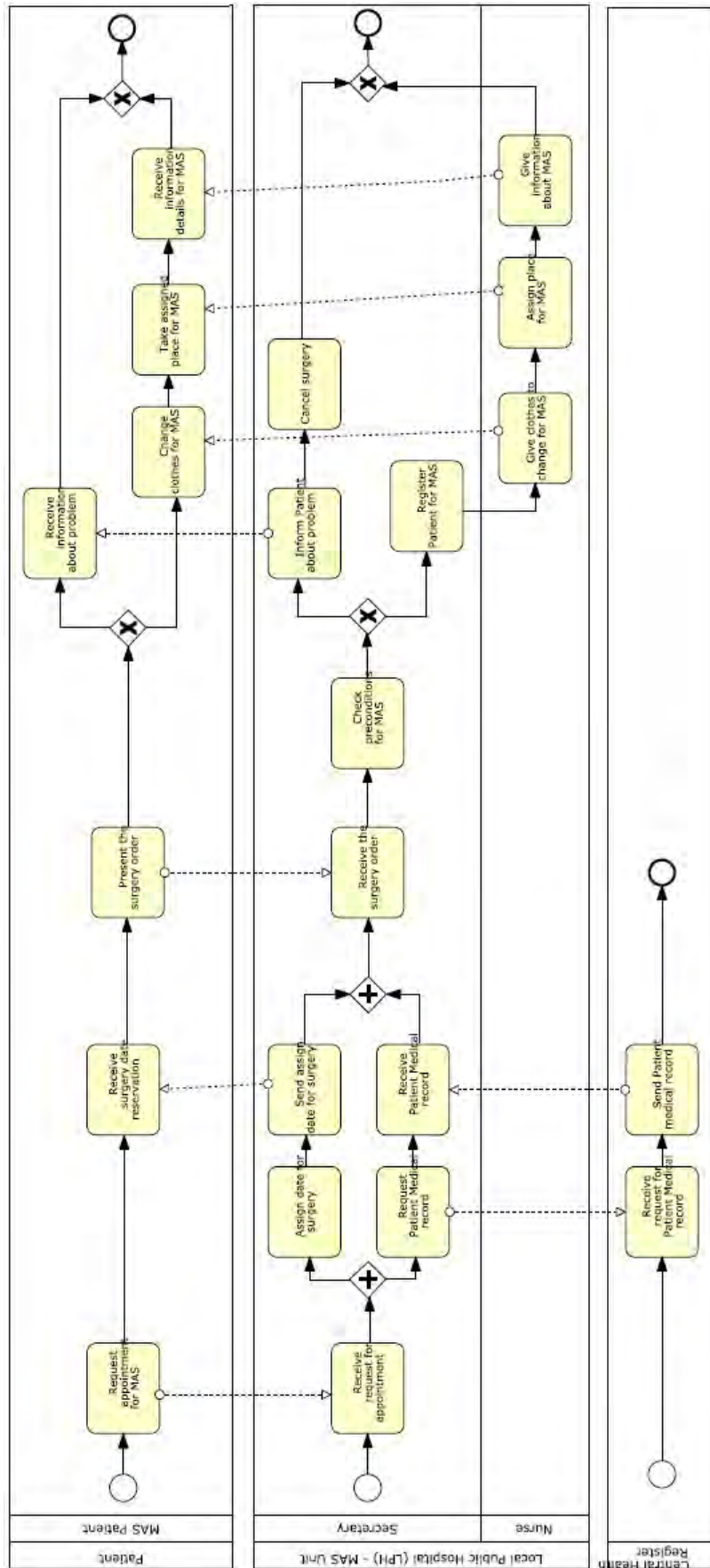


Figure 10.25.: Adapted Patient MAS in BPMN2 modeled in Oryx



Each pool in the BP is modeled as a substitution transition that in turn is modeled in a subpage of the hierarchy defined. The Hospital subpage models the petri net corresponding to the Hospital in which each activity is also modeled as a substitution transition. This allows us to log the three times set for the execution of the activities in the BP: enabled, start and complete times in each subpage.

In Figure 10.26 the subpage for the Hospital model is presented, showing the transitions substitution approach to model each activity defined as a substitution transition that is detailed on its own page, in which we can log the three times defined for each activity. We use this approach also as a way to organize the petri net model as if we were adding the log of the three times defined for each activity in a single page, the model would have become into a web of transitions, places and inscriptions for the MXML logging.

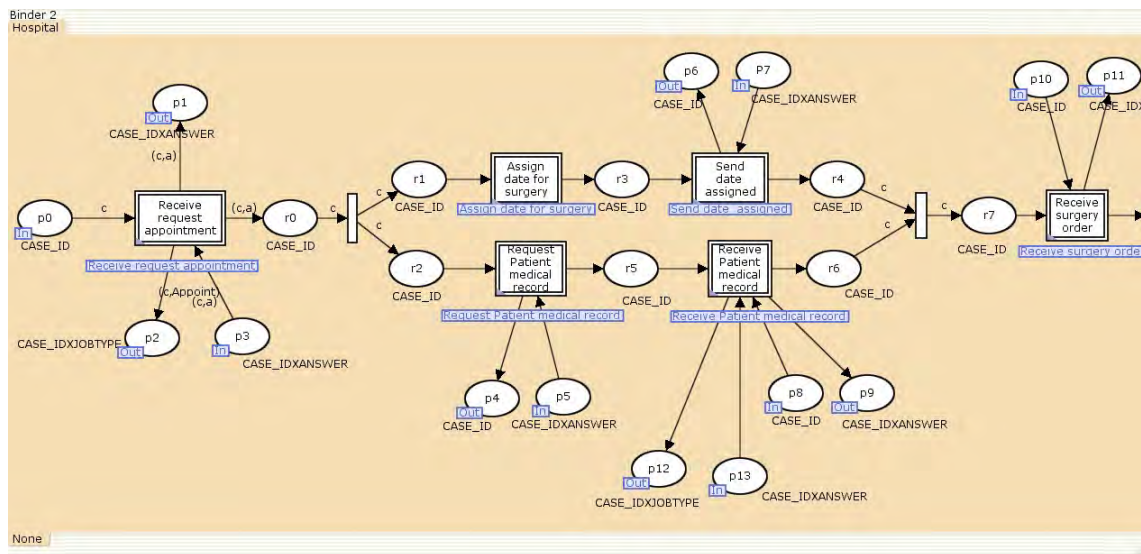


Figure 10.26.: Hospital sub-page showing the transitions substitution modeling

The resources modeling approach we use is based on the one proposed in [Rozinat et al., 2008] to simulate the realization of each activity by a person assigned to the roles of the BP, as a centralized place that is shared between all pages to simulate the availability of resources between activities. We have added the “system” resource for executing the automated activities realized by services, and the “service” resource for executing services. As the resources are modeled in a centralized (fusion) place that is shared by all activities in the BP, the availability of resources is simulated in a more realistic way, as it is divided between BP cases and activities instances. In Figure 10.27 the resources modeling used is shown for the top page environment in which the BP cases are started and the central fusion place for resources is defined.

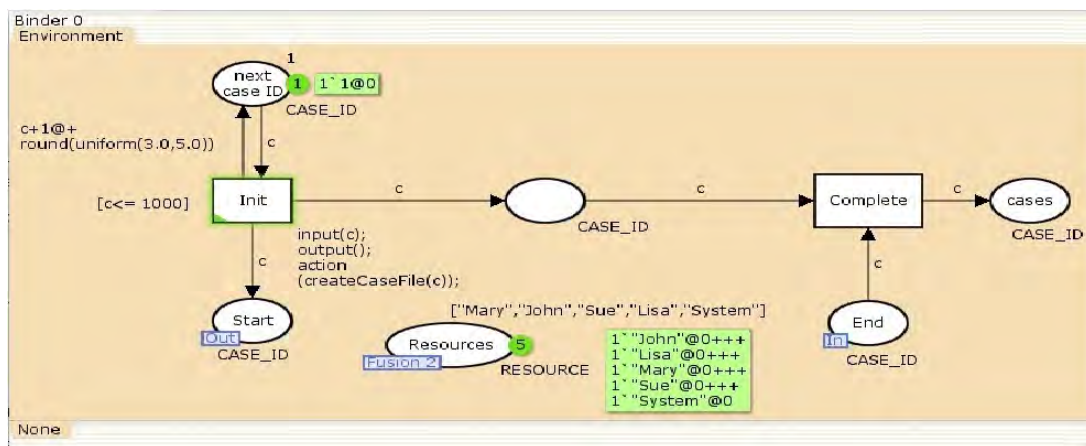


Figure 10.27.: Top page environment for resources modeling with central fusion place

In Figure 10.28 the resources modeling is shown for the “Check preconditions for MAS” activity, as an example of what is modeled for each activity defined in the BP. The centralized (fusion) place allows resources to execute each activity when they became available, returning to it once the execution of the activity has finished. The definition of the three execution times to be logged for each activity in the BP model along with the inscriptions needed for the ProM log generation in the MXML format can also be seen.

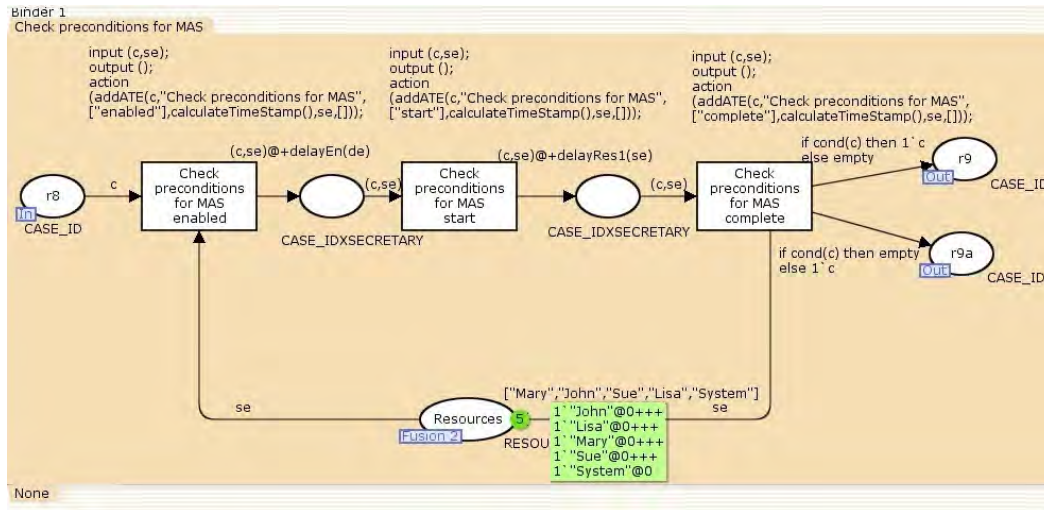


Figure 10.28.: Resources modeling example for activity “Check preconditions for MAS”

A more complete approach for resources modeling is proposed in [van der Aalst et al., 2010] but we decided not to use it for this simulation as it introduces more complexity for the modeling of the BP. Nevertheless, the modeling tried for this alternative can be seen in Appendix E. For services execution modeling we use a queuing approach for each service executing in the Server, in order to emulate the real execution of several services in the same infrastructure. We define the services resources to attend the execution of the services invoked, running simulations with several different configurations for the delay in each queue, the maximum of jobs to be enqueued at each time, the time for processing each service invoked and the result of the execution. In Figure 10.29 the service subpage is presented, showing the modeling of the two services offered by the HGCR.

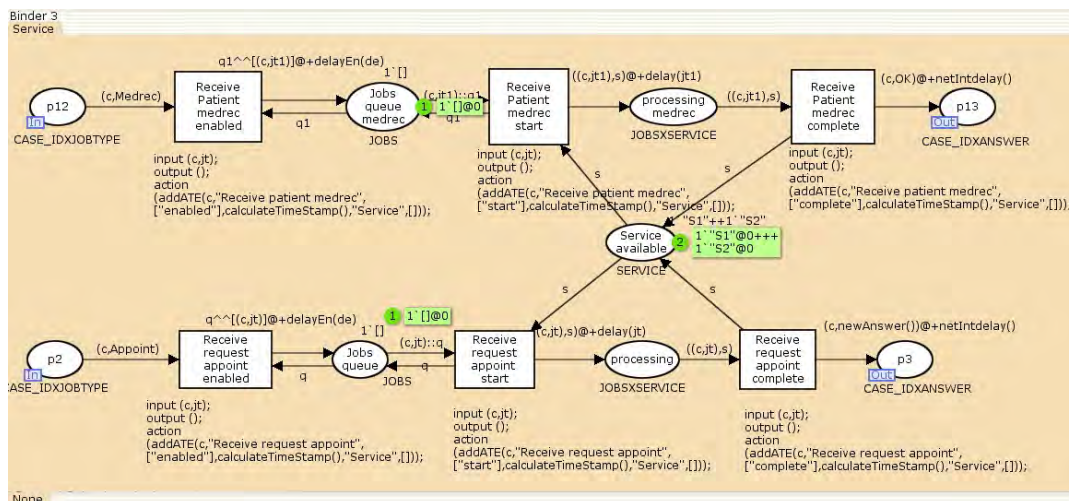
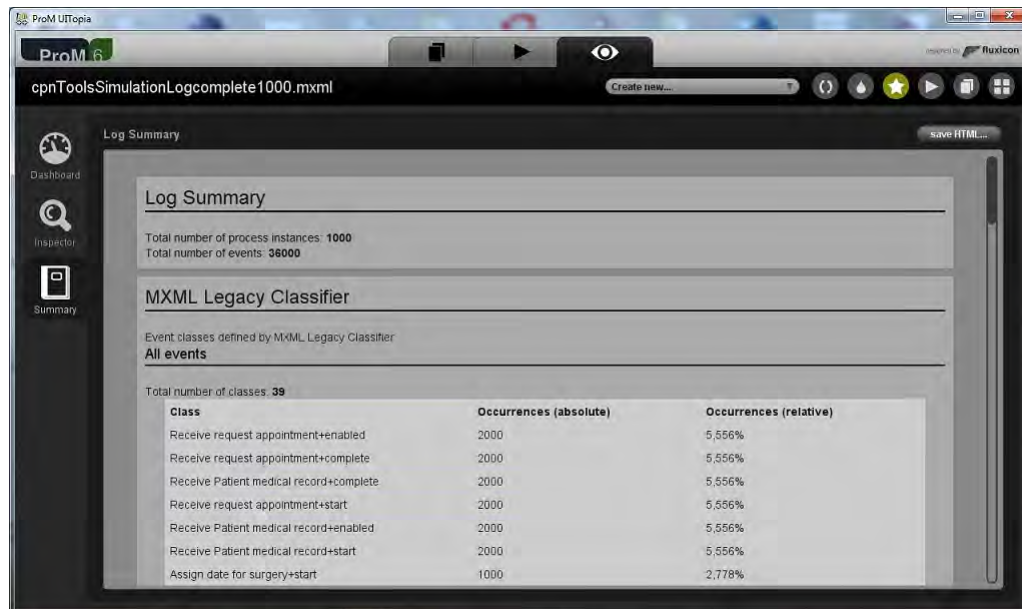


Figure 10.29.: Service sub-page with a queuing approach

The steps carried out for the simulation preparation can be seen as corresponding to the implementation of the measures in the process engine in the Configuration phase, including the ProM log inscriptions to generate the event logs for calculating the execution measures. Specific aspects of petri nets and CPN Tools modeling, simulation and ProM event logs can be seen in [CPN Group, Alves de Medeiros and Guenther, 2005, van der Aalst and Stahl, 2011].

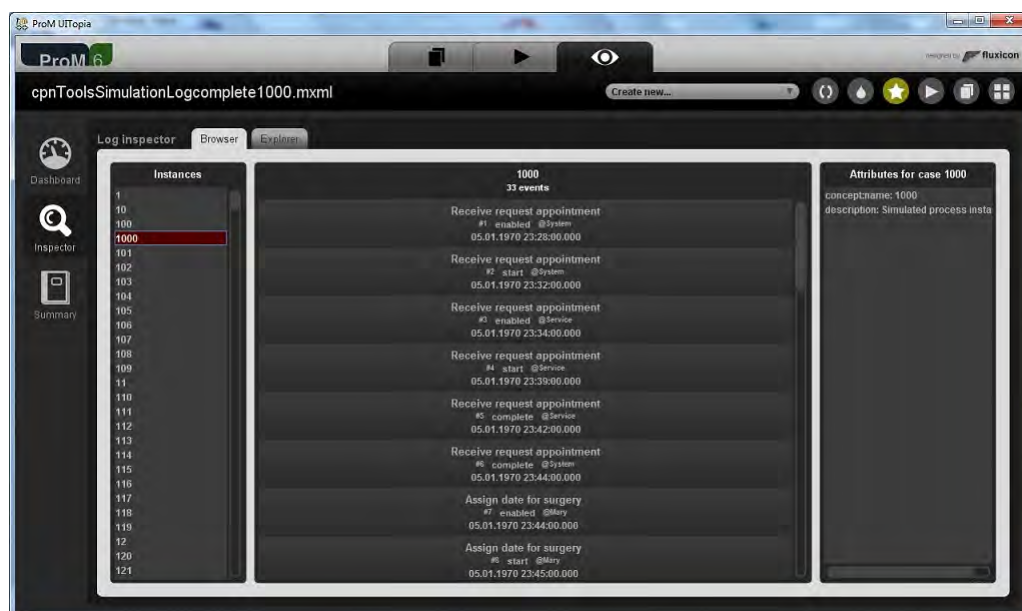
### 10.4.8.3. Analysis with ProM

The simulation performed corresponds to the execution of the BP cases in the Execution phase of BPCIP. In the Evaluation phase, once the event logs corresponding to the simulated execution are obtained we can load them into the BPEMM ProM plug-in to calculate and visualize the execution measures defined. In Figure 10.30 the summary of the event log in the ProM framework is shown.



**Figure 10.30.:** Summary of the event log from CPNTools loaded into ProM

As the CPN Tool creates a log for each case of the execution of the simulation, we use the ProMimport framework to merge the logs generating one MXML file containing the execution of the total BP cases run in the simulation, which were a thousand BP cases. In Figure 10.31 the inspector option of ProM shows the execution of each BP instance.



**Figure 10.31.:** Inspector option of ProM showing the execution of BP instances

Although the improvements activities have to be performed by the business people, as the BP was not really implemented in the organization, here we provide an example of how the activities should be carried out to find improvements for the BP, integrating these to generate a new version

of the BP to be executed again so as to compare the new execution results with the ones from the previous version.

Recalling that the BP execution Average Throughput Time (TT) goal was defined by business people to be under 90 minutes, the Warning rank between 90 and 120 minutes and the Problems rank to be above 120 minutes, these times should be analyzed to see whether they are in accord with the definitions or not. This analysis corresponds to the execution of the EM4 - Analyze execution measurement results activity. The ProM BPEMM plug-in execution measures are shown in Figure 10.32 for all BP cases with average times, where it can be seen that the Average TT for all the BP cases is above the goal defined.



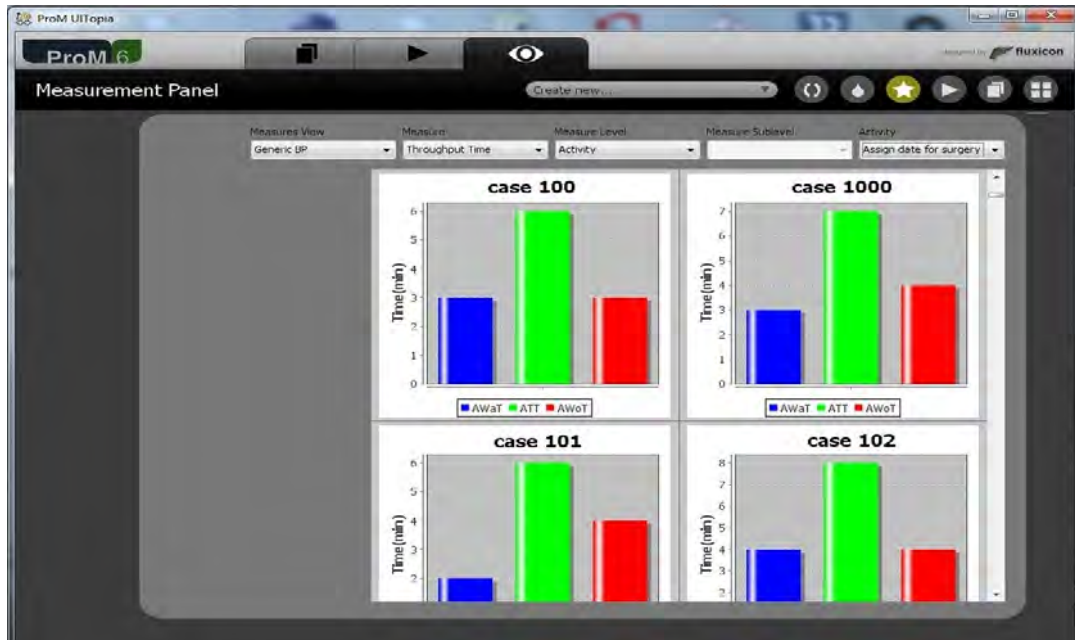
Figure 10.32.: ProM BPEMM plug-in time measures for all BP cases

In Figure 10.33 the times for each BP case (selected BP case 999) and each activity executed in the case with total times and percentages times as defined by the indicators are shown.



Figure 10.33.: ProM BPEMM plug-in time measures for each BP case and its activities

It can be seen that the TT for this particular case is above the goal defined, and the waiting time for the Activity “Assign date for Surgery” is almost twice of the working time for the same activity. It would be interesting then to drill down to the times for the Activity through all BP cases, to analyze the execution of the activity in all BP cases, finding that in in most cases its waiting time is greater than or equal to its working time, as shown in Figure 10.34.



**Figure 10.34.:** ProM BPEMM plug-in time measures for the activity “Assign date for Surgery”

This leads to the finding of an improvement opportunity referring to the definition of the activity “Assign date for surgery”. On analyzing the BP it is found that this activity is performed manually by the persons assigned to the Secretary role, who have to take the activity from their work list and assign a suitable day and hour to carry out the surgery, by looking in the calendar for available surgery slots. This is not the only activity in which the persons participate, so the times between actions could be long.

Once the improvement opportunities are found the improvement activities are executed in the Evaluation phase, to define and plan the modification of the BP in the next BP lifecycle execution, starting with the BP model to see if there is a suitable redesign that allows the activity and/or related activities to be changed. In the Define improvements activity, the improvement to be integrated to the activity “Assign date for surgery” is specified, its goal being to lower the activity waiting time to fifty percent of its current time. The Diagnose BPs activity is not executed as the BP we are dealing with is not the original one from the organization and the real value of this activity is to detect organizational improvements for the definition of the BP.

In the Formulate improvements activity the need to evaluate several redesign alternatives for the activity and related ones is specified. After the activities and associated documentation to support the improvement effort are executed in the Evaluation phase, the Design&Analysis phase is executed again, to evaluate the redesigns for the BP model in the activity BM3 - Redesign BPs, as shown in Figure 10.35. From the possible redesigns heuristics in [Reijers, 2003] the Task Composition (COMPOS) and Task Automation (AUTO) ones are combined to obtain one automated activity to assign and send the surgery date.

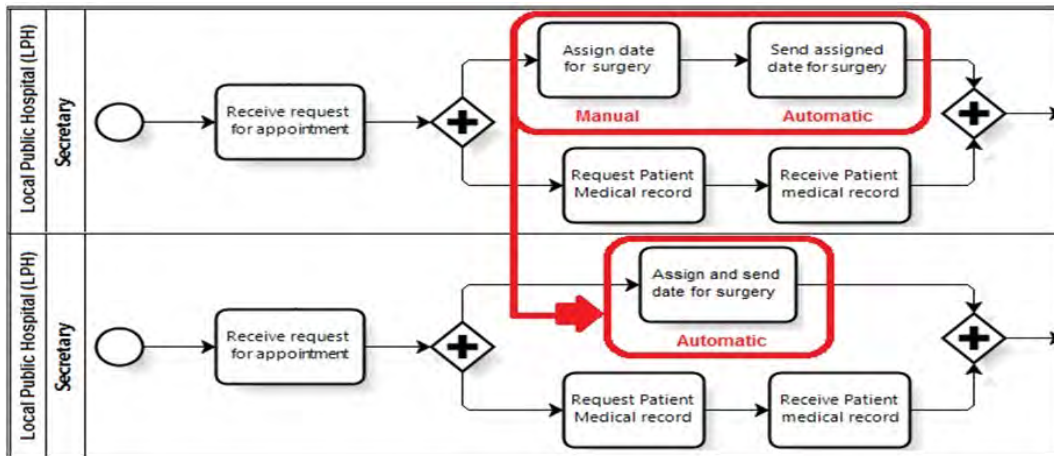


Figure 10.35.: Redesign options for the activity “Assign surgery date”

In the Configuration phase the new version of the BP is modified to support the changes made, in this case study we modeled another service in the CPNTools Petri net to perform the assignation of the date for the surgery as well as to send the information to the patient. In addition, the measures implementation is adjusted to register the data needed from the new BP version adding the corresponding data for the changed activity and the new service.

In the Execution phase the new version is simulated again (in a real environment it would be executed again in the process engine) and the event logs are generated. In the Evaluation phase the event log with the information of the simulation of the new BP version is loaded in the BPEMM ProM plug-in to calculate and visualize the selected measures. In Figure 10.36 the new values for the resulting combined activity are shown.

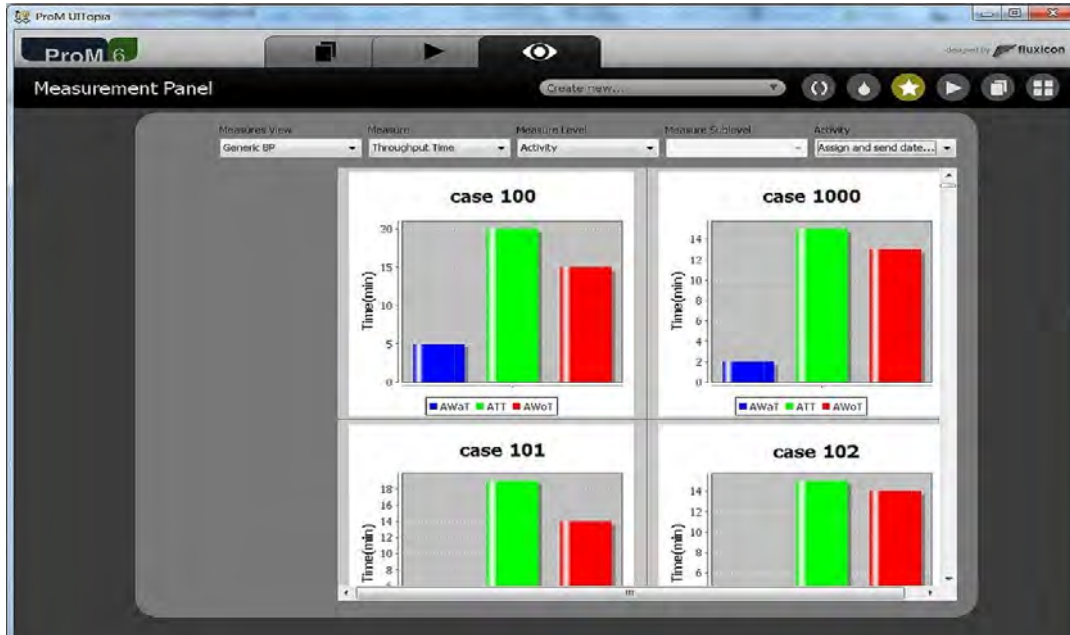


Figure 10.36.: BPEMM ProM plug-in measures for the new activity

The values obtained would be compared to the BP goals, the improvement goals for the effort as well as to the values from the execution of the previous version of the BP, to see whether the introduction of the improvement has actually led to an improvement in the execution of the BP. In this case we can see that this is the case, but as it was all based on a simulation of the BP, the actual times are not the important thing to be evaluated, but rather the feasibility of the BPCIP

proposal, which can be assessed as feasible and useful as it allows us to analyze and compare the execution of the BPs, which was the objective.

#### 10.4.9. Conclusions and lessons learned

Based on the implementation and execution performed in the three process engines selected in the pre-production environment defined, as well as the simulation in CPNTools, the Configuration and Execution phases activities and artifacts defined are feasible for putting into operation in organizations for implementation and execution of their BPs. The tool support is validated as the three process engines allow the implementation and execution of the Patient MAS BP as defined by the HGCR, including the registration of execution data and its extraction to be loaded into the ProM framework and the ProM BPEMM plug-in to perform the analysis of the BPs execution.

Based on the analysis presented of the interviews with the Responsible of the quality group and the BP, the proposals in BPCIP are feasible to be incorporated in organizations wanting to manage and improve their BPs, and the execution measures provided by BPEMM that would be added to the existing ProM plug-in prototype would allow us to calculate and visualize several existing indicators for the BP and also provide more information for the analysis of the BPs execution.

As regards lessons learned, this case study showed us that it takes time and effort to set up the environment for BPs execution, at least with the process engines selected, but once everything is in place the implementation and execution of BPs can be done with successful results, even in these tools which are all free or community edition. This shows that the focus on BPs is gaining in importance as several community efforts exist to provide such platforms, and apart from this, many non-evaluated commercial tools are available that incorporate the support needed for executing BPs.

The interoperability from BP engines with the ProM framework was assessed and successfully carried out, by means of the chain of outputs/inputs from the .csv file extracted from the process engine, transforming it with ProMImport and/or Fluxicon into MXML format and loading it in the ProM framework. One issue detected that needs to be improved is the interoperability from BP modelers to BP engines; BPMN2 is gaining in importance, however, since the standard also includes the semantic definitions for the execution of BPs, as well as the standard interchange format in XML, so we believe this will soon be solved.

The ProM BPEMM plug-in although a prototype for now, allows us to analyze the execution measurement results from BPs execution, presenting information for all BP cases executed, each BP case executed and the corresponding activities, and each activity in all BP cases, presenting the different levels of granularity defined to provide the needed insight into BPs execution.

### 10.5. Conclusions

In this chapter the validation of the MINERVA framework has been presented divided in three empirical validations: an experiment and two case studies. The aim of the experiment was to assess the Suitability of the QVT transformations defined and the Understandability of the SoaML service models generated, for which the corresponding information about the planning, design, operation, analysis and interpretation of data was described, along with examples of the experimental material. The experiment allowed us to conclude that the QVT transformations defined are suitable and its results are understandable with high percentages of agreements, although more work has to be done to be able to generalize the results.

A case study was defined initially to be carried out completely in the HGCR hospital in Ciudad Real, Spain, but due to difficulties in implementing and executing the prototype of the BP in the hospital, it was split into two case studies: one case study to assess BPCIP in general (without BPSOM) working with business people from the hospital and defining a pre-production environment (laboratory) to implement, execute and simulate the BP, and another case study to assess

BPSOM and the automatic generation of services working with a software development team from ANTEL in Montevideo, Uruguay.

The design, protocol, execution, collection and analysis of data for each case study was presented, explaining the adaptations that had to be performed for each one, and presenting elements and screenshots from the different tools used as defined and presented in chapter 9. Both case studies allow us to assess the feasibility of the proposals of BPCIP and BPSOM. These were evaluated by the involved participants and seen as useful by them, although more work needs to be done to be able to generalize the results. The tool support defined was also evaluated allowing the successful realization of the activities defined.



## Chapter 11.

# Conclusions and future work

This Chapter presents the conclusions of the results of this thesis work. In the first place in section 11.1 the attainment of objectives is analyzed based on the main results achieved, after which the publications obtained from this thesis work are presented in section 11.2 and the main research contributions in section 11.3; finally open research lines and future work are discussed in section 11.4.

### 11.1. Attainment of objectives

The analysis of the attainment of objectives is presented first based on the results achieved for each partial objective defined in chapter 1, and then the attainment of the overall objective is presented, based on these.

- **PO.1:** Study the paradigms BPM, SOC and MDD and the main current proposals related with the application of SOC and MDD to BPM.
  - The study of paradigms BPM, SOC and MDD is presented in the state of the art in chapter 3.
  - The study of the main current proposals related to the application of SOC and MDD to BPM was carried out as a systematic review of the literature, also presented in chapter 3.
- **PO.2:** Study the main standards for languages and metamodels to: (1) model BPs and services and (2) represent the execution of collaborative BPs.
  - The study of the main standards for languages and metamodels to model BPs and services is presented in chapter 3 including the selected BP modeling language BPMN2 and service modeling language SoaML.
  - The study of the main standards for languages and metamodels to represent the execution of collaborative BPs is presented in chapter 3 including XPDL, WS-BPEL and the recent BPMN2 executable version.
- **PO.3:** Define the overall structure and elements to be integrated in the framework.
  - The overall structure and elements integrated in MINERVA framework are presented in chapter 4 providing the definition of the framework by means of a Dimensions view comprising the conceptual, methodological and tools support dimensions, and the Process view defining the BPCIP lifecycle and method of work of MINERVA
- **PO.4:** Adapt and integrate into the framework a service-oriented methodology for service systems development from BPs.
  - A previously-existing service-oriented methodology for service systems development from BPs was adapted and integrated in the framework although the version presented in chapter 7 is the newest one (the previous version is referred in the publications section 11.2).

- the service-oriented methodology was named BPSOM and extended with the use of BPMN2 for BP modeling, SoaML for services modeling and the QVT transformations defined between them, which is presented in chapter 7, including the use of BPMN2 in the corresponding activity of the Business Modeling Discipline and the use of SoaML in the corresponding activities of the Design Discipline, along with the integration and use of the defined QVT transformations for SoaML service models generation from BPMN2 models.
- **PO.5:** Study and define concepts and relationships (ontology) and transformations between BP models and service models.
  - Several existing standards were studied and analyzed to define an ontology for BP and service modeling, which is presented in chapter 4 and is part of a larger ontology which is also presented in chapter 4, envisioned for the support of the complete BP lifecycle.
  - Based on the definition of the ontology for BP and service modeling QVT transformations were defined to automatically generate SoaML service models from BPMN2 models, which are presented in chapter 8, including the correspondences between elements in both metamodels and the defined rules to transform elements from BPMN2 to SoaML.
- **PO.6:** Define the tools support for carrying out service oriented development from BPs with a model-driven approach, and implement prototypes.
  - The Eclipse MINERVA design distribution was defined integrating several existing Eclipse plug-ins and frameworks such as EMF, and developing an Eclipse SoaML plug-in to provide support for SoaML modeling, as well as an Eclipse iS4BP plugin to insert the invocation of services into BPMN2, XPDL and WS-BPEL models, which is presented in chapter 9.
- **PO.7:** Adapt and integrate a continuous process improvement approach into the framework focusing on BPs execution measurement.
  - Activities from an existing continuous process improvement approach were selected, adapted and integrated into the framework with a focus on BPs execution measurement, which is presented in chapter 5, including the integration of improvements activities in the Evaluation phase of BPCIP, the focus on execution measurement activities throughout the BP lifecycle and their realization through it.
- **PO.8:** Define a set of execution measures for the execution of BPs implemented by services.
  - BPEMM was defined integrating a set of execution measures for the execution of BPs implemented by services as a key part of the continuous process improvement approach which is presented in chapter 6, including the three views for Generic BP, Lean and Services, the dimensions of time, cost, quality and flexibility and the hierarchy defined.
- **PO.9:** Define techniques and tools for the analysis of the execution of BPs implemented by services, and implement prototypes.
  - The ProM framework was selected as process mining techniques provider, for the analysis of the execution of BPs implemented by services, and a BPEMM ProM plug-in (prototype) was developed to provide support for the analysis of BPEMM execution measures, which is presented in chapter 9.
- **PO.10:** Validate the proposals of the framework by means of experiments (generation of services) and case studies.
  - The QVT transformations defined for the generation of SoaML service models from BPMN2 models were validated empirically with an experiment to assess the suitability (of the QVT transformations for the generation of SoaML models from BPMN2 models) and the understandability (of the QVT transformations results i.e. the SoaML models generated) sub-characteristics as defined in ISO 9126, which is presented in chapter 10.

- The proposal was validated by means of two case studies focusing on different parts of MINERVA framework: one to assess the feasibility of BPCIP and BPEMM in the context of a project with the Hospital General de Ciudad Real (HGCR), and the other to assess the newest version of BPSOM including the use of BPMN2, SoaML and the automatic generation of SoaML models from BPMN2 models in the context of the entity of telecommunications from the Uruguayan government, ANTEL, which are both presented in chapter 10.

In Table 11.1 a summary of the attainment of objectives is presented, along with the chapter in which the objective is further developed.

**Table 11.1.:** Relation Partial Objectives - Chapters in this thesis

Partial Objective	Chapters
PO.1	3
PO.2	3
PO.3	4
PO.4	7
PO.5	4,8
PO.6	9
PO.7	5
PO.8	6
PO.9	9
PO.10	10

On the basis of the results presented for the attainment of partial objectives, these were successfully achieved, in this author's opinion.

Recalling that the overall objective of this thesis work was stated in chapter 1 as:

***To define a framework to provide support for the continuous improvement of business processes based on SOC and MDD.***

on the basis of the attainment of partial objectives this objective was also achieved successfully achieved in this author's opinion, such that the research hypothesis for this thesis work:

***New paradigms such as Service Oriented Computing (SOC) and Model Driven Development (MDD) can offer support to set up in organizations a continuous improvement cycle for their business processes.***

is positively confirmed in this author's opinion.

## 11.2. Results that support this thesis

Several of the proposals and results of this thesis have been published in different scientific forums, which are summarized in Table 11.2 differentiated by type of publication (Journal, Conference, Book, etc.) and scope of the scientific forum (International, Iberoamerican <sup>1</sup>). Publications that are currently submitted for evaluation are denoted as (+).

---

<sup>1</sup>Iberoamerican: includes Latin America, Spain and Portugal.

**Table 11.2.:** Summary of publications from this thesis

Type of publication	International	Iberoamerican	Total
Journal articles (JCR)	+2		+2
Journal articles (other)	1		1
Book chapters	1		3
Conferences level A	1		1
Conferences level B	5		5
Conferences level C	0		0
Other Conferences	2	5	7
Workshops	2	4	6
Total	12+2	9	21+2

All publications listed in Table 11.2 are peer-reviewed, Journal articles (JCR) correspond to Journals as indexed in the Journal Citation Reports of Thomson Reuters (former ISI) <sup>2</sup>, Journal articles (other) correspond to journals not indexed, the ranking of Conferences corresponds to the one as defined in the ERA<sup>3</sup> ranking, Workshops not ranked are listed separated, as are non-ranked Conferences.

For the complete list of publications their classification by the particular Partial Objective any given one describes is presented in Table 11.3, where the code for the publication corresponds to the name of the Journal, Conference, Book or Workshop plus the year of publication. This code is used in the next sections to provide the complete information for each publication.

**Table 11.3.:** Publications by Partial Objectives

Publication	PO.1	PO.2	PO.3	PO.4	PO.5	PO.6	PO.7	PO.8	PO.9	PO.10
CCIS-12a, ICSOF-10	X									
JSI-10, WESOA-09			X							
CAISE-11, JCIS-11, BPSC-09, IDEAS-09				X						
RCIS-10, JIISIC-10, PNIS-09a					X					
SAC-12, ICSTE-10, DSDM-10					X					
CCIS-12b& ENASE-11, JISBD-11a, PNIS-10, PNIS-09b, IGI-09							X	X		
MOSE-10, JISBD-11b						X				
IST-12					X	X				X
JASIST-12							X	X	X	X

In the following the publications are listed grouped by type of publication as defined in Table 11.2 and the corresponding code as presented in Table 11.3.

<sup>2</sup>[http://thomsonreuters.com/products\\_services/science/science\\_products/a-z/journal\\_citation\\_reports/](http://thomsonreuters.com/products_services/science/science_products/a-z/journal_citation_reports/)

<sup>3</sup>[http://www.arc.gov.au/xls/ERA2010\\_conference\\_list.xls](http://www.arc.gov.au/xls/ERA2010_conference_list.xls)

### 11.2.1. Journal articles (JCR)

**IST-12:** Delgado, A., Garcia-Rodríguez de Guzmán, I., Ruiz, F., Piattini, M., Model-driven development of Service-oriented systems: generation of SoaML service models from BPMN 2.0 business process models, In Information and Software Technology (IST), 2012, submitted.

**JASIST-12:** Delgado, A., Weber, B., Ruiz, F., Garcia-Rodríguez de Guzmán, I., Piattini, M., An integrated approach based on execution measures for the continuous improvement of business processes realized by services, Journal of the American Society for Information Science and Technology (JASIST), 2012, submitted.

### 11.2.2. Journals articles (other)

**JSI-10:** [Delgado et al., 2010h] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M. “A Model-driven and Service-oriented framework for the business process improvement”, In Journal of Systems Integration, Vol.1, No. 3, July 2010. Internet Journal of Systems Integration ISSN: 1804-2724 <http://www.si-journal.org/index.php/JSI/index>. Extended article from WESOA'09.

### 11.2.3. Book Chapters

**IGI-09:** [Sánchez González et al., 2009] Sánchez González, L., Delgado, A., Ruiz, F., García, F., Piattini, M., “Measurement and Maturity of Business Processes”. Eds.: Cardoso, J. and van der Aalst, W., Handbook of Research on Business Process Modeling, Ed. 1, London UK / PA USA, Information Science Reference (IGI Global), 2009. ISBN: 978-1-60566-288-6 pp. 532-556.

### 11.2.4. Conferences Level A

**CAISE-11:** [Delgado et al., 2011c] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Business Process Service Oriented Methodology (BPSOM) with Service generation in SoaML”. In: 23rd International Conference on Advanced Information Systems Engineering (CAISE'11), June 2011, Londres, UK. Acceptance ratio 14.6875%.

### 11.2.5. Conferences Level B

**SAC-12:** [Delgado et al., 2012a] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Model Transformations for Business-IT Alignment: From Collaborative Business Process to SoaML Service Model”, In: 27th Symposium On Applied Computing (SAC'12), March 2012, Riva del Garda (Trento), Italy. Acceptance ratio TBA.

**CCIS-12a:** [Delgado et al., 2012b] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Main principles on the integration of SOC and MDD paradigms to business processes: a systematic review”, In Communications in Computer and Information Science (CCIS) series, Springer-Verlag, In press, 2012. Extended selected article of best papers from ICSOFT'10.

**ICSOFT-10:** [Delgado et al., 2010g] (Delgado et al., 2010) Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Application of service-oriented computing and model-driven development paradigms to business processes: a systematic review”, In: 5th International Conference on Software and Data Technologies (ICSOFT'10), Atenas, Greece, Julio 2010. Acceptance ratio 9.4% for full papers (30.8% others).

**CCIS-12b:** [Delgado et al., 2012c] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Continuous improvement of business processes realized by services based on execution measurement”, In Communications in Computer and Information Science (CCIS) series, Springer-Verlag, In press, 2012. Selected article of best papers from ENASE'11. Also published as **ENASE-11:** [Delgado et al., 2011e] Delgado, A., Weber, B., Ruiz, F., García-Rodríguez de Guzmán, I.,

“Execution measurement-driven continuous improvement of Business processes implemented by services”. In: 6th International Conference on Evaluation of Novels Approaches to Software Engineering (ENASE’11), June 2011, Beijing, China. Acceptance ratio 32.7%.

**RCIS-10:** [Delgado et al., 2010f] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Towards an ontology for service oriented modeling supporting business processes”, In: IV International Conference on Research Challenges in Information Science (RCIS’10), Niza, Francia, May 2010, IEEE publication, ISBN 978-1-4244-4840-1, Acceptance ratio 45.27%.

### 11.2.6. Other Conferences

#### International

**ICSTE-10:** [Delgado et al., 2010b] Delgado, A., García-Rodríguez de Guzmán, I., Ruiz, F., Piattini, M., “From BPMN business process models to SoaML service models: a transformation-driven approach”. In: 2nd International Conference on Software Technology and Engineering (ICSTE 2010), San Juan de Puerto Rico, Puerto Rico, USA, October 2010. Not ranked Conference.

**BPSC-09:** [Delgado et al., 2009b] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “Towards a Service-Oriented and Model-Driven Framework with Business processes as first-class citizens” In: 2nd International Conference on Business Process and Services Computing (BPSC’09), Leipzig, Germany, March 2009. Lecture Notes in Informatics ISBN 978-3-88579-2413, pp. 19-31.

#### Iberoamerican

**JCIS-11:** [Delgado et al., 2011a] Delgado, A., García-Rodríguez de Guzmán, I., Ruiz, F., “Desarrollo de servicios con SoaML desde procesos de negocio en BPMN: metodología y automatización”, In: VII Jornadas de Ciencia e Ingeniería de Servicios (JCIS’11), Setiembre 2011, A Coruña, España.

**JISBD-11a:** [Delgado et al., 2011b] Delgado, A., González, L., Larroca, S., Pastorini, A., Ruiz, F., García-Rodríguez de Guzmán, I., “SoaML Eclipse plug-in para modelado de servicios”, Eclipse SoaML plug-in demo In: XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD’11), Setiembre 2011, A Coruña, España.

**JISBD-11b:** [Delgado et al., 2011d] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Weber, B., “Marco MINERVA para mejora continua de procesos de negocio implementados con servicios”. In: XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD’11), Setiembre 2011, A Coruña, España.

**JISIC-10:** [Delgado et al., 2010d] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., “Ontología para el ciclo de vida de los procesos de negocio implementados con servicios”. In: Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JISIC’10), Noviembre 2010, Mérida, México.

**IDEAS-09:** [Delgado et al., 2009a] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., “Desarrollo de software orientado a servicios basado en procesos de negocio”, In: XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software (IDEAS’09), Abril 2009, Medellín, Colombia -from 2010 CibSE Congreso Iberoamericano en "Software Engineering"-. ISBN: 978-958-44-5028-9, pp.1-14.

### 11.2.7. Workshops

#### International

**MOSE-10:** [Delgado et al., 2010a] Delgado, A., García-Rodríguez de Guzmán, I., Ruiz, F., Piattini, M., “Tool support for service oriented development from business processes”, In: 2nd. International Workshop on Model-Driven Service Engineering (MoSE’10), 48th. International Conference on Objects, Models, Components, Patterns (TOOLS’10), Málaga, Spain, Junio 2010. Internet

Proceedings of 2nd MoSE 2010, CEUR Workshop Proceedings, ISSN 1613-0073, <http://CEUR-WS.org/Vol-608/paper3.pdf>.

**WESOA-09:** [Delgado et al., 2009c] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., Piattini, M., “MINERVA: Model drIveN and sErvice oRIented framework for the continuous business process improVement & relAted tools”, In: V Workshop on Engineering Service-Oriented Applications (WESOA’09), International Conference on Service Oriented Computing (ICSOC’09), Stockholm, Sweden, November 2009. Services Science Subline, LNCS, Springer Verlag.

### Iberoamerican

**DSDM-10:** [Delgado et al., 2010c] Delgado, A., García-Rodríguez de Guzmán, I., Ruiz, F., Piattini, M., “Generación de modelos de servicios en SoaML desde modelos de procesos de negocio en BPMN con QVT”, In: VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM’10) en XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD’10), Setiembre 2010, Valencia, España.

**PNIS-10:** [Delgado et al., 2010e] Delgado, A., Ruiz, F., García-Rodríguez de Guzmán, I., “Mejora continua de procesos de negocio basada en PmCompetisof integrando BPMM”. In: III Taller sobre Procesos de Negocio e Ingeniería de Servicios (PNIS’10) en XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD’10), Setiembre 2010, Valencia, España.

**PNIS-09a:** [Delgado and García-Rodríguez de Guzmán, 2009] Delgado, A., García-Rodríguez de Guzmán, I., “Ontología para relacionar procesos de negocio y su realización como servicios”. In: II Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS’09) en XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD’09), Setiembre 2009, San Sebastián, España.

**PNIS-09b:** [Sánchez González and Delgado, 2009] Sánchez González, L., Delgado, A., “Medidas para Procesos de Negocio y su alineamiento en BPMM”. In: II Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS’09) en XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD’09), Setiembre 2009, San Sebastián, España.

## 11.3. Main research contributions

The main research contributions of this PhD thesis to the different research areas that the MINERVA framework tackles are as follows:

- the complete MINERVA framework proposal integrates the existing business and IT visions in organizations, providing methodological guides and tool support for the management and continuous improvement of BPs. The elements from the methodologies have been defined to provide a systematic way of developing service-oriented systems from BPs, for the execution measurement of both BPs and services and for the integration of the improvement opportunities found, in a formal, yet non-complex way. The stakeholders from both the business and IT areas can thus follow the guides provided, where the activities, work products and roles defined are as few as possible and hence easy to learn and apply with the tool support defined. What is more, the Business Process Service Oriented Methodology (BPSOM) defines specific elements for service-oriented development, to be added to the existing software development process used in the organization. The aim is to reuse the existing culture, knowledge and way of working of the organization as much as possible. The Business Process Continuous Improvement Process (BPCIP) also minimizes the elements for guiding the execution measurement and improvement effort, proposing four activities for each discipline defined, with the aim of its being rapidly understood and used by both business and IT people.
- BPCIP, and the Business Process Execution Measurement Model (BPEMM) that it integrates, focus on the business area of the organization. Apart from the methodological guide and tool support -contribution, BPEMM integrates a set of existing execution measures and new ones that have been defined, which covers many of the measures that organizations usually define to gather information about the real execution of their BPs. By applying

QQM, it is defined in such a way that each measure is related to a specific goal of the organization; it can be extended by adding new goals and associated questions and measures, if an organization needs to do so. The approach integrates different methods, techniques and proposals, which makes it easier use it as a guide for the execution effort in organizations, as all concepts, information and definitions are together in one single place.

- BPSOM, and the MDA approach it integrates are focused on the IT area of the organization. Apart from the methodological guide and tool support contribution, the automatic generation of SoaML service models from BPMN2 models, by means of QVT transformations, provides a way to reuse knowledge about the correspondences between BP models and service model elements. This, in turn, helps maintain traceability between elements from both models, to analyze the impact of changes, and to allow code from different platforms to be generated as desired, from the service models generated.
- for the tools support provided, we have maximized the integration of existing free tools as much as possible, which allows us and MINERVA users to extend them as needed. When no tool existed that provided the functionalities needed, we developed our own, such as the Eclipse SoaML plug-in for service modeling with the SoaML standard, and the Eclipse iS4BP-e plug-in for inserting the invocation of services generated into the BPMN2, XPD L and WS-BPEL files for BP execution in a suitable process engine. The development of service-oriented systems from BPs is supported by our own Eclipse distribution, called Eclipse MINERVA design, which integrates all the needed plug-ins. For the evaluation of BPs implemented by services, we have developed another plug-in, named ProM BPEMM plug-in, in this case for the ProM framework, which implements the calculation and visualization of BPEMM execution measures. Although it is a prototype at the moment, it provides the basis for analyzing the execution of BPs based on the BP Generic view and the time dimension.

Other contributions of the research carried out in this PhD thesis include:

- the seven principles found as a result of the systematic literature review carried out, highlighting the main aspects that have to be taken into account when integrating the SOC and MDD paradigms to BPs: BPs modeling, service oriented modeling, model transformations, methodological approach, use of patterns, collaborative processes and tool support. This could be useful for other researchers working in these areas, providing a research context.
- the BPs and service modeling sub-ontologies, which highlight the main concepts involved in the modeling of both BPs and services, as well as the relationships between the two areas. These relationships were the basis for the definition of the correspondences between BPMN2 and SoaML metamodels, which in turn led us to the definition of the QVT transformations for the automatic generation of SoaML service models from BPMN2 models. This could be also useful for researchers working in these areas, providing a basis for further reasoning.

## 11.4. Open research lines and future work

Based on the development and results obtained from this thesis work, several open research lines were identified and are described below, which could be the subject of future work:

- Complete the initial sketch of the ontology defined to support the BP lifecycle as this author believes that the conceptualization of the elements involved in each of the defined phases will help towards a better understanding of these and thus lead to better support for BP management and improvement in organizations.
- Extend the QVT transformations to take into account other constructions in BPMN2 models that could be also identified as services of coarser granularity (see chapter 3) than the ones we generate, as a service can realize an activity, a sub-process or an entire BP. In addition, other constructions such as process patterns could be suitable for generation of services to support more complex interactions in the BP.



- Asses the possibility of also extending the QVT transformations to include QoS (Quality of Service) modeling on the SoaML service model generated using the QoS profile by OMG, to add information on the expected execution of services to be taken into account in their implementation.
- Extend the Eclipse SoaML plug-in or develop another plug-in in order to generate code from the SoaML service model we now obtain, to be able to close the chain from BP modeling to BP implementation with our own tools in the Eclipse environment for development, making the SoaML model an output/input artifact in the software development, as the BPMN2 model is already from the business area to the software area.
- Extend the BPEMM, to be able to manage data for other activities states such as suspend and resume, adding other measures of interest in the dimensions in each execution view, as well as for the ones for which we have not specified execution measures in this first version of BPEMM, as we have left these open to be extended as future work.
- Complete the BPEMM ProM plug-in so that it calculates and shows the complete set of execution measures from BPEMM, providing business people with the means for a complete analysis of the execution of BPs implemented by services.
- Carry out more case studies to apply BPCIP, BPCIP with BPSOM and BPSOM completely in appropriate organizations, to asses the perceived benefits of the proposals that were highlighted by the people from the case studies presented, and to improve the necessary ones.
- Carry out several replications of the QVT transformations experiment, to be able to provide more confidence in the results of the assessment of their suitability and understandability.



# Appendices



## Appendix A.

# Data extraction from primary studies of the systematic review

### A.1. Overview

This Appendix presents a summary of the data extraction forms used in the systematic literature review presented in chapter 3, to retrieve the relevant data from the primary studies selected. The complete forms used also contain information on each specific principle for the integration of paradigms, indicating whether the proposal includes it or not, and in which way. As the data corresponding to the principles is presented in chapter 3 it is not repeated here, and only the summary of the proposal is presented.

### A.2. Data extraction tables

The tables that are presented in the following includes: the corresponding reference for the study, the title, authors and their filiations, a brief summary of the proposal and where it is published. The tables are presented all together and without numbering for the sake of clarity.

Reference	(Hu et al., 2003)
Title	Conceptual Framework and Architecture for Service Mediating Workflow Management
Authors	Jinmin Hu, Paul Grefen
Filiations	Computer Science Department, University of Twente, The Netherlands
Summary	A three level conceptual framework is defined to relate business process with implemented services, adding a service mediating layer, and defining five related layers. A Service Invocation Coordinator (SIC) is defined to implement service invocation.
Published	Information and Software Technology, Volume 45, Issue 13, (2003)

Reference	(Baghdadi, Y. 2004)
Title	ABBA: an architecture for deploying business-to-business electronic commerce applications
Authors	Youcef Baghdadi
Filiations	Department of Mathematics and Computer Science, UAE University Al-Ain, United Arab Emirates
Summary	A four level architecture and a design process to develop B2B applications are defined. It proposes to select the services to realize business process from an existing repository, registry or catalog.
Published	Electronic Commerce Research and Applications, Vol.3, Is.2, (2004)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Liew et al, 2004)
Title	A Framework for Business Model Driven Development
Authors	Philip Liew <sup>1</sup> , Kostas Kontogiannis <sup>1</sup> , Tack Tong <sup>2</sup>
Filiations	<sup>1</sup> Dept. of Electrical and Computer Engineering, University of Waterloo, Canada <sup>2</sup> IBM Canada IBM Toronto Laboratory, Canada
Summary	UML software artifacts such as UML AD, use cases, collaboration and deployment diagrams are obtained from annotated BPMN BP models enhanced with required information to transform them.
Published	Proceedings of the International Workshop on Software Technology and Engineering Practice (STEP'04)

Reference	(Henkel et al, 2005)
Title	Supporting Development and Evolution of Service-based Processes
Authors	Martin Henkel <sup>1</sup> , Jelena Zdravkovic <sup>2</sup>
Filiations	<sup>1</sup> Stockholm University and Royal Institute of Technology, Sweden, <sup>2</sup> University of Gävle and Royal Institute of Technology, Sweden
Summary	Defines a set of transformation patterns, called realization types, to transform a business process into a technical process based on existing services provided by internal systems, defining levels to identify the quality of the transformation between them.
Published	Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'05)

Reference	(Quartel et al, 2005)
Title	An approach to relate business and application services using ISDL
Authors	Dick Quartel, Remco Dijkman and Marten van Sinderen
Filiations	Centre for Telematics and Information Technology, University of Twente, The Netherlands
Summary	A service oriented design approach is proposed, to relate services modeled in different levels of abstraction: business and application, with techniques based on ISDL: profiles to relate models and model conformance.
Published	Proceedings of the 9th IEEE International EDOC Enterprise Computing Conference (EDOC'05)

Reference	(Chen et al, 2006)
Title	A Generative Business Process Prototyping Framework
Authors	Ang Chen, Didier Buchs
Filiations	Computer Science Department, University of Geneva, Switzerland
Summary	Defines a methodological framework with a language based on Petri Nets for modeling, verification and prototyping of business processes, along with an approach to obtain services from business processes.
Published	Proceedings of the 17th IEEE International Workshop on Rapid System Prototyping (RSP'06)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(de Castro et al, 2006)
Title	A model driven method for service composition modelling: a case study
Authors	Valeria de Castro, Esperanza Marcos and Marcos López Sanz
Filiations	Kybele Research Group, Department of Software and Computing Systems, Rey Juan Carlos University, Spain
Summary	The main focus is the model-driven development of service oriented Web Systems defining models, metamodels and transformations between them to obtain a service composition model which expresses the interaction of services to perform business processes, and a methodology to guide the development.
Published	Web Engineering and Technology, Vol. 2, No. 4, (2006)

Reference	(Mendling et al, 2006)
Title	Transformation of yEPC Business Process Models to YAWL
Authors	Jan Mendling, Michael Moser, Gustaf Neumann
Filiations	Vienna University of Economics and Business Administration, Austria
Summary	Horizontal transformations between BP modeling notations from yEPC to YAWL which allows to analyze models with verification tools. BP modeled in yEPC and YAWL showing transformations between them.
Published	Proceedings of the 21st. Symposium on Applied computing (SAC'06)

Reference	(Mili et al, 2006)
Title	Classifying Business Processes for Domain Engineering
Authors	Hafedh Mili, Mohand Frendi, Guitta Bou Jaoude, Louis Martin, Guy Tremblay
Filiations	Department of Computer Science, Université du Québec à Montréal, Canada
Summary	Method to develop information systems from generic BP assembling sw components corresponding to defined parts of the models. Represent and classify generic BP in four views: informational, functional, dynamical and organizational using UML.
Published	Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)

Reference	(Murzek et al, 2006)
Title	Structural Patterns for the Transformation of Business Process Models
Authors	Marion Murzek, Gerhard Kramler, Elke Michlmayr
Filiations	Vienna University of Technology, Austria
Summary	Model transformation approach based on domain specific patterns to obtain horizontal transformation, model integration and synchronization between BP modeling notations. BP modeled in ADONIS and EPC showing transformations between the notations.
Published	Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Orriens et al, 2006)
Title	A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaboration
Authors	Bart Orriens <sup>1</sup> , Jian Yang <sup>2</sup>
Filiations	<sup>1</sup> Department of Information Systems And Management Tilburg University, Tilburg, The Netherlands <sup>2</sup> Department of Computing Macquarie University, Sydney, Australia
Summary	Proposes a business rules driven approach for the development of adaptive collaborative service oriented business processes. It provides a framework for the development of business collaborations with three dimensional views: collaboration aspects, business and technical requirements.
Published	Proceedings of the IEEE International Conference on Services Computing (SCC'06)

Reference	(Papazoglou et al, 2006)
Title	Service-oriented design and development methodology
Authors	Michael P. Papazoglou, Willem-Jan van den Heuve
Filiations	INFOLAB, Department of Information Systems and Management, Tilburg University, The Netherlands
Summary	Defines phases, activities and artifacts for the development of services associated with business processes. It differs from ours in that although it defines guides for a service oriented development, it is focused on the implementation of services as WS, adding technical aspects that cannot be used with other technologies.
Published	Web Engineering and Technology, Vol. 2, No. 4, (2006)

Reference	(Roser et al, 2006)
Title	Model- and Architecture-Driven Development in the Context of Cross-Enterprise Business Process Engineering
Authors	Stephan Roser <sup>1</sup> , Bernhard Bauer <sup>1</sup> , Jörg P. Müller <sup>2</sup>
Filiations	<sup>1</sup> Institute of Computer Science, University of Augsburg, Germany <sup>2</sup> Institute of Computer Science, Clausthal University of Technology, Germany
Summary	Models and metamodels for services are defined to relate them to business processes and the underlying architecture, focusing the derivation of services on three architectures: brokerless, centralized and decentralized broker, providing a technical focus.
Published	Proceedings of the IEEE International Conference on Services Computing (SCC'06)

Reference	(Sadiq et al, 2006)
Title	Model Driven Distribution of Collaborative Business Processes
Authors	Wasim Sadiq <sup>1</sup> , Shazia Sadiq <sup>2</sup> , Karsten Schulz <sup>1</sup>
Filiations	<sup>1</sup> SAP Research Centre Brisbane, Australia <sup>2</sup> School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, Australia
Summary	Service oriented design approach to relate services modeled at different abstraction levels such as business and application. Automatic distribution of collaborative BPs from an integrated one, and merge of various BPs into one.
Published	Proceedings of the IEEE International Conference on Services Computing (SCC'06)



Appendix A. Data extraction from primary studies of the systematic review

Reference	(Tao Tao et al, 2006)
Title	Develop Service Oriented Finance Business Processes: A Case Study in Capital Market
Authors	Aries Tao Tao, Jian Yang
Filiations	Department of Computing, Macquarie University, Sydney, Australia
Summary	Methodology for developing services from BP defining steps for BP analysis, design and implementation based on SOA, services are identified from essential functionalities of the system modelled in generic way.
Published	Proceedings of the IEEE International Conference on Services Computing (SCC'06)

Reference	(Zhao et al, 2006)
Title	Supporting Virtual Organisation Alliances with Relative Workflows
Authors	Xiaohui Zhao, Chengfei Liu and Yun Yang
Filiations	Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
Summary	Provides support for BP collaborations of dynamic virtual organizations on the basis of a service-oriented relative workflow, adding definitions for IT and BP, where private information is hidden according to visibility constraints.
Published	Proceedings of the 3rd. Asia-Pacific Conference on Conceptual Modelling (APCCM'06)

Reference	(Tao Tao et al, 2007)
Title	Supporting Differentiated Services With Configurable Business Processes
Authors	Aries Tao Tao, Jian Yang
Filiations	Department of Computing, Macquarie University, Sydney, Australia
Summary	Approach to deliver differentiated services realized by configurable BP to achieve service flexibility, manageability and reusability. core BP model in FSM (Finite state machine) to specify generic activities applicable in all circumstances and separate others applicable to a particular group of people (as usage context) transformed into CCBP (configurable context BP).
Published	Proceedings of the IEEE International Conference on Web Services (ICWS'07)

Reference	(Zdun et al, 2007)
Title	Modeling Process-Driven and Service-Oriented Architectures Using Patterns and Pattern Primitives
Authors	Uwe Zdun <sup>1</sup> , Carsten Hentrich <sup>2</sup> , Schahram Dustdar <sup>1</sup>
Filiations	<sup>1</sup> Vienna University of Technology, Austria <sup>2</sup> CSC Deutschland Solutions GmbH, Germany
Summary	Defines patterns to guide the definition, transformation and implementation of technical processes using software services from business processes in which they call process-driven service oriented architecture.
Published	ACM Transactions on the Web, Vol. 1, No. 3, (2007)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Bruckmann et. al, 2008)
Title	AMABULO - A Model Architecture for Business Logic
Authors	Tobias Bruckmann, Volker Gruhn
Filiations	University of Leipzig, Applied Telematics / e-Business Group, Germany
Summary	Maps stereotypes actions to user and system functions in a defined metamodel described in an XML schema as interchange format and input for transformation engines, modeling BPs with UML AD, and software systems with UML.
Published	Proceedings of the 15th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'08).

Reference	(Cauvet et. al, 2008)
Title	Business Process Modeling: a Service-Oriented Approach
Authors	Corine Cauvet, Gwladys Guzelian
Filiations	LSIS - Université Paul Cézanne (Aix-Marseille III), France
Summary	Defines an iterative service composition process in which services matching BP requirements are selected and alternative services can be generated, using ontologies to match BP requirements to service goals.
Published	Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS'08)

Reference	(Chen, H., 2008)
Title	Towards Service Engineering: Service Orientation and Business-IT Alignment
Authors	Hong-Mei Chen
Filiations	Department of Information Technology Management, Shidler College of Business, University of Hawaii, Manoa
Summary	Defines the BITAM-SOA framework for business-IT alignment, defining three layers comprising specific modules, using them to guide a process model for service design, development and management, which can be top-down or bottom-up.
Published	Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS'08)

Reference	(de Castro et al, 2008)
Title	A model driven approach for the alignment of Business and Information Systems Models
Authors	Valeria de Castro, Juan Manuel Vara Mesa, Elisa Herrmann, Esperanza Marcos
Filiations	Kybele Research Group, Department of Software and Computing Systems, Rey Juan Carlos University, Spain
Summary	They integrate a business value model to the proposal in (de Castro et. al, 2006) from which to derive software artifacts, specifically adding the business view and models, and transformation from them to use case model.
Published	Proceedings of the Mexican International Conference on Computer Science (ENC'08)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Gacitua-Decar et al,2008)
Title	Pattern-based business-driven analysis and design of service architectures
Authors	Veronica Gacitua-Decar and Claus Pahl
Filiations	School of Computing, Dublin City University, Glasnevin, Ireland
Summary	A pattern based technique is used in a layered architecture defined in a framework for the designing of EI architectures, for service identification, transformation from business models to service architecture, among others. Process, domain and SOA patterns are identified which are implemented and organized in pattern catalogues, and can be instantiated, suggested to apply compared, modified and merged.
Published	Proceedings of the 3rd International Conference on Software and Data Technologies (ICSOFIT'08)

Reference	(Herold et. al, 2008)
Title	A Seamless Modeling Approach for Service-Oriented Information Systems
Authors	Sebastian Herold <sup>1</sup> , Jan Ebell <sup>2</sup> , Andreas Rausch <sup>1</sup> , Christian Linsmeier <sup>2</sup> , Alexander Bösl <sup>3</sup> , Detlef Peters <sup>3</sup>
Filiations	<sup>1</sup> Clausthal University of Technology, Germany, <sup>2</sup> Josef Witt GmbH, Weiden, Germany, <sup>3</sup> MID GmbH, Nuremberg, Germany
Summary	Proposes a model driven approach to relate business process with software services in a target (distributed) three layered architecture. The business process are modeled in UML deriving an analysis model which is then mapped to elements in the design model defined in the existing target architecture, and then to the implementation model.
Published	Proceedings of the 5th International Conference on Information Technology: New Generations (ITNG'08)

Reference	(Huemer et. al, 2008)
Title	Inter-organizational Systems: From Business Values over Business Processes to Deployment
Authors	Christian Huemer <sup>1</sup> , Philipp Liegl <sup>1</sup> , Rainer Schuster <sup>2</sup> , Hannes Werthner <sup>2</sup> , and Marco Zapletal <sup>2</sup>
Filiations	<sup>1</sup> Business Informatics Group, <sup>2</sup> Electronic Commerce Group, Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria
Summary	Define three layers based on the Open-edi reference model for inter-organizational systems: business operational view (BOV) comprising business models and BP model layers, and the functional service view (FSV) comprising a deployment artifacts layer.
Published	Proceedings of the 2nd IEEE International Conference on Digital Ecosystems and Technologies (DEST'08)

Reference	(Oquendo, F., 2008)
Title	Formal Approach for the Development of Business Processes in terms of Service-Oriented Architectures using $\pi$ -ADL
Authors	Flavio Oquendo
Filiations	European University of Brittany, University of South Brittany, France
Summary	Defines mappings between BPM constructs and PI-ADL for SOA expressions, based on process patterns and BPMN core elements for obtaining services
Published	Proceedings of the IEEE International Symposium on Service-Oriented System Engineering (SOSE'08)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Rychly et. al, 2008)
Title	Modeling of service oriented architecture: from business process to service realization
Authors	Marek Rychly, Petr Weiss
Filiations	Faculty of Information Technology, Brno University of Technology, Czech Republic
Summary	Proposes modeling of business process realization by services diagrams, identifying services, and integrating business process modeling and object modeling by means of a Business Services Model (BSM), a mediator between requirements and implementation.
Published	Proceedings of the 3rd International Conference on Evaluation of Novel Approaches to Software Engineering, (ENASE'08)

Reference	(Thomas et. al, 2008)
Title	Using Process Models for the Design of Service-Oriented Architectures: Methodology and E-Commerce Case Study
Authors	Oliver Thomas <sup>1</sup> , Katrina Leyking <sup>1</sup> , Florian Dreifus <sup>2</sup>
Filiations	<sup>1</sup> Institute for Information Systems (IW <sub>i</sub> ) at the German Research Center for Artificial Intelligence (DFKI), Saarland University, Germany <sup>2</sup> SAP Inc., Germany
Summary	Define horizontal transformations between EPC conceptual model to BPMN conceptual-technical model, and vertical ones from BPMN to BPEL for process execution.
Published	Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS'08)

Reference	(Bai et al., 2009)
Title	A Service-Oriented Business Process Modeling Methodology and Implementation
Authors	Lin Bai, Jun Wei
Filiations	Institute of Software, Chinese Academy of Sciences, Beijing, China
Summary	Defines six steps with activities to navigate from a BPMN model which is remodeled with execution information, to a BPEL implementation.
Published	Proceedings of the International Conference on Interoperability for Enterprise Software and Applications (I-ESA'09)

Reference	(Brambilla et al., 2009)
Title	Model-Driven Engineering of Service Orchestrations
Authors	Marco Brambilla, Matteo Dosmi, Piero Fraternali
Filiations	Dipartimento di Elettronica e Informazione Politecnico di Milano, Milano, Italy
Summary	Top-down model-driven approach defining several steps from going to BPMN models to implementation using WS and Web interfaces.
Published	Proceedings of the IEEE Congress on Services-I (SERVICES'09)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Delgado et al., 2009)
Title	MINERVA: Model drIveN and sErvice oRIented Framework for the Continuous Business Process improVement and relAted Tools
Authors	Andrea Delgado <sup>1</sup> , Francisco Ruiz <sup>2</sup> , Ignacio García-Rodríguez de Guzmán <sup>2</sup> , and Mario Piattini <sup>2</sup>
Filiations	<sup>1</sup> Computer Science Institute, Faculty of Engineering, Montevideo, Uruguay <sup>2</sup> Technologies and IS Depto., Faculty of Computer Science, University of Castilla-La Mancha, Spain
Summary	Framework MINERVA defined in three dimensions: conceptual, methodological and tools support, along with several elements defined including an initial service-oriented methodology, and proposals for the automatic derivation of services from BPs.
Published	Proceedings of the 5th International Workshop on Engineering Service-Oriented Applications (WESOA'09) in (ICSOC'09)

Reference	(Kohlborn et. al, 2009)
Title	Identification and Analysis of Business and Software Services-A Consolidated Approach
Authors	Thomas Kohlborn <sup>1</sup> , Axel Korthaus <sup>1</sup> , Taizan Chan <sup>2</sup> , and Michael Rosemann <sup>1</sup>
Filiations	<sup>1</sup> Business Process Management Group, Information Systems Cluster, Faculty of Science and Technology, Queensland University of Technology, Australia <sup>2</sup> School of Information Technology, Faculty of Science and Technology, Queensland University of Technology, Australia
Summary	A survey on thirty existing service development approaches is presented, along with a consolidated approach proposal in which two main parts for the process are defined: the derivation of business services and the derivation of software services to support them.
Published	IEEE transactions on services computing, Vol. 2, No. 1, (2009)

Reference	(Norton et al., 2009)
Title	Towards the Ontology-based Transformation of Business Process Models
Authors	Barry Norton
Filiations	STI Innsbruck, University of Innsbruck, Austria
Summary	Chain of transformations involving ontologies to go from sBPMN and sEPC models to BPMP models, being a common abstraction for BP modeling, from which to generate sBPEL models using WSML to effect this transformation.
Published	Proceedings of the 4th International Workshop on Semantic Business Process Management (SBPM'09) in (ESWC'09)

Reference	(Patig et al., 2009)
Title	Role of Process Modeling in Software Service Design
Authors	Susanne Patig <sup>1</sup> , Harald Wesenberg <sup>2</sup>
Filiations	<sup>1</sup> University of Bern, IWI, Switzerland <sup>2</sup> StatoilHydro ASA, Norway
Summary	An hybrid service design process modeling BPs from which to identify services in a top-down manner and also a bottom-up identification from information concepts and existing application systems.
Published	Proceedings of the 7th International Conference on Service Oriented Computing (ICSOC'09)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Touzi et. al, 2009)
Title	A model-driven approach for collaborative service-oriented architecture design
Authors	Jihed Touzi1, Frederick Benaben1, Herve Pingaud1, Jean Pierre Lorre2
Filiations	1 Ecole des Mines d'Albi-Carmaux, France 2 EBM Web Sourcing, parc technologique du canal, Ramonville StAgne, France
Summary	Proposes a model driven approach but for collaborative service oriented architecture, to transform BPMN models into UML models and BPEL models. It defines a collaborative SOA metamodel composed of three views: service, information and process; the business process model is transformed to each one.
Published	Production Economics, Volume 121, Issue 1, (2009)

Reference	(Weber et al., 2009)
Title	Towards a Methodology for Semantic Business Process Modeling and Configuration
Authors	Ingo Weber <sup>1</sup> , Jorg Hoffmann <sup>2</sup> , Jan Mendling <sup>3</sup> , and Jorg Nitzsche <sup>4</sup>
Filiations	<sup>1</sup> SAP Research Karlsruhe, Germany, <sup>2</sup> University of Innsbruck, DERI, Austria <sup>3</sup> BPM Cluster, Queensland University of Technology, Australia, <sup>4</sup> Institute of Architecture of Application Systems, University of Stuttgart, Germany
Summary	A methodology is proposed for the modeling and configuration of BP adding a semantic approach to implement services from BP.
Published	Proceedings of the 2nd International Workshop on Business-Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (SeMSoC'07) in (ICSOC'07) publication 2009.

Reference	(Dahman et al., 2010)
Title	Generation of Component Based Architecture from Business Processes: Model Driven Engineering for SOA
Authors	Karim Dahman, Francois Charoy, Claude Godart
Filiations	Universite de Lorraine, UHP - LORIA, France
Summary	Uses BPMN conversations to model participants and messages exchanged to be transformed into SCA models.
Published	Proceedings of the 8th IEEE European Conference on Web Services (ECOWS'10)

Reference	(Delgado et al., 2010)
Title	From BPMN business process models to SoaML service models: a transformation-driven approach
Authors	1Andrea Delgado, 2Ignacio García-Rodríguez de Guzmán, 2Francisco Ruiz, 2Mario Piattini
Filiations	1Computer Science Institute, Faculty of Engineering, University of the Republica, Montevideo, Uruguay 2 Alarcos Research Group, Technology and IS Department, University of Castilla – La Mancha, Ciudad Real, Spain
Summary	MDA approach for generating service models in SoaML from collaborative BP models in BPMN by means of QVT transformations, obtaining services from activities in different pools which correspond to participants.
Published	Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE'10)

Appendix A. Data extraction from primary studies of the systematic review

Reference	(Elvesaeter et al., 2010)
Title	Aligning Business and IT Models in Service-Oriented Architectures using BPMN and SoaML
Authors	Brian Elvesaeter <sup>1</sup> , Dima Panfilenko <sup>2</sup> , <sup>3</sup> Sven Jacobi & Christian Hahn
Filiations	1 SINTEF ICT, Norway, 2 DFKI IWi, Germany, Saarstahl, Germany
Summary	Proposes transformations from BPMN to SoaML using ATL, modeling BPs including business rules and data, for EPC and BPMN models to SoaML models.
Published	Proceedings of the 1st Workshop on Model Driven Interoperability (MDI'2010) in (MODELS'10)

Reference	(Lazarte et al., 2010)
Title	Model-Driven Development Methodology for B2B Collaborations
Authors	Ivanna M. Lazarte <sup>1</sup> , Edgar Tello-Leal <sup>1,2</sup> , Jorge Roa <sup>1</sup> , Omar Chiotti <sup>1,3</sup> and Pablo D. Villarreal <sup>1,3</sup>
Filiations	1CIDISI, Universidad Tecnologica Nacional-FRSF, Santa Fe, Argentina 2Universidad Autonoma de Tamaulipas, Tamaulipas, Mexico 3INGAR-CONICET, Santa Fe, Argentina
Summary	A top-down model-driven approach with four phases: Bussiness Analysis, Design of Business solution, Design of IT Architecture solution and Design of the Technological solution, with a global view on B2B process obtaining system interfaces in the desired technology.
Published	Proceedings of the International Workshop on Models and Model-driven Methods for Service Engineering (3M4SE'10) in (EDOC'10)

Reference	(Sinha et al., 2010)
Title	Use Cases to Process Specifications in Business Process Modeling Notation
Authors	Avik Sinha, Amit Paradkar
Filiations	IBM T J Watson Research Center, Hawthorne, NY, USA
Summary	Transformations from use case models to BPMN models, synchronizing the requirements definition.
Published	Proceedings of the 8th IEEE International Conference on Web Services (ICWS'10)





## Appendix B.

# BPCIP and BPSOM Web Sites (implemented with EPF Composer)

### B.1. Overview

In this Appendix the Web Sites implemented with EPF Composer for BPCIP<sup>1</sup> and BPSOM<sup>2</sup> are presented, as they are online in the Web site of this thesis. The structure of each site is organized by means of the defined Disciplines, activities, roles, work products and guidelines, along with the defined phases for each lifecycle.

### B.2. BPCIP Web Site

Once in the BPCIP Site the Introduction page is displayed, presenting a general description of BPCIP and its use, as shown in Figure B.1. On the left side, the structure of the site is shown as a tree: in the first place the BPCIP introduction page; below it the Disciplines folder containing one folder for each discipline defined; then the Work products folder also containing one folder for each discipline with the corresponding artifacts; after that the Roles folder containing the roles defined, then the Lifecycle folder showing the defined phases, and finally the Guidance folder containing the templates defined for the artifacts.



Figure B.1.: BPCIP Introduction page

<sup>1</sup><http://alarcos.esi.uclm.es/MINERVA/BPCIP/Published/>

<sup>2</sup><http://alarcos.esi.uclm.es/MINERVA/BPSOM/Published/>

## B.2.1. Disciplines

In the following, a screenshot of an activity definition is presented as an example for each Discipline in BPCIP, where on the left side all the activities defined for the Discipline are displayed, and on the right side, the definition for the selected activity is shown.

This definition provides in the first place, a description of the activity and the purpose defined for doing it, and then in the Relationships box, a summary of the elements it comprises with the corresponding links to their description are presented: Roles (primary and additional performers), Inputs (mandatory and optional) and Output artifacts for the activity, and in the Process usage, the use of the activity in the defined lifecycle.

### B.2.1.1. Business Modeling Discipline

The Business Modeling Discipline presents the description and elements for the activities defined in it: BM1-Asses the organization, BM2-Identify Business Processes and BM3-Redesign Business Processes. In Figure B.2 the page for the BM3-Redesign Business Processes activity is shown as an example.

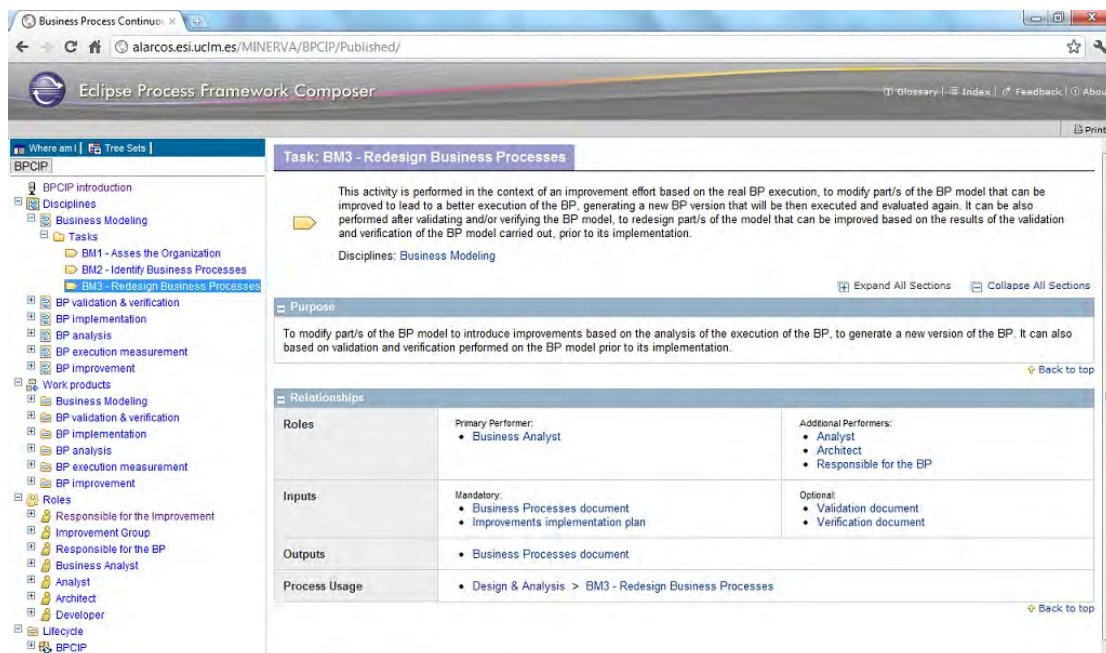


Figure B.2.: BM3-Redesign Business Processes example

### B.2.1.2. BP Validation and Verification Discipline

The BP Validation and Verification Discipline presents the definition and description of the activities it comprises: VV1-Validate Business Processes and VV2-Verify Business Processes, from which the first one is presented in Figure B.3 as an example.

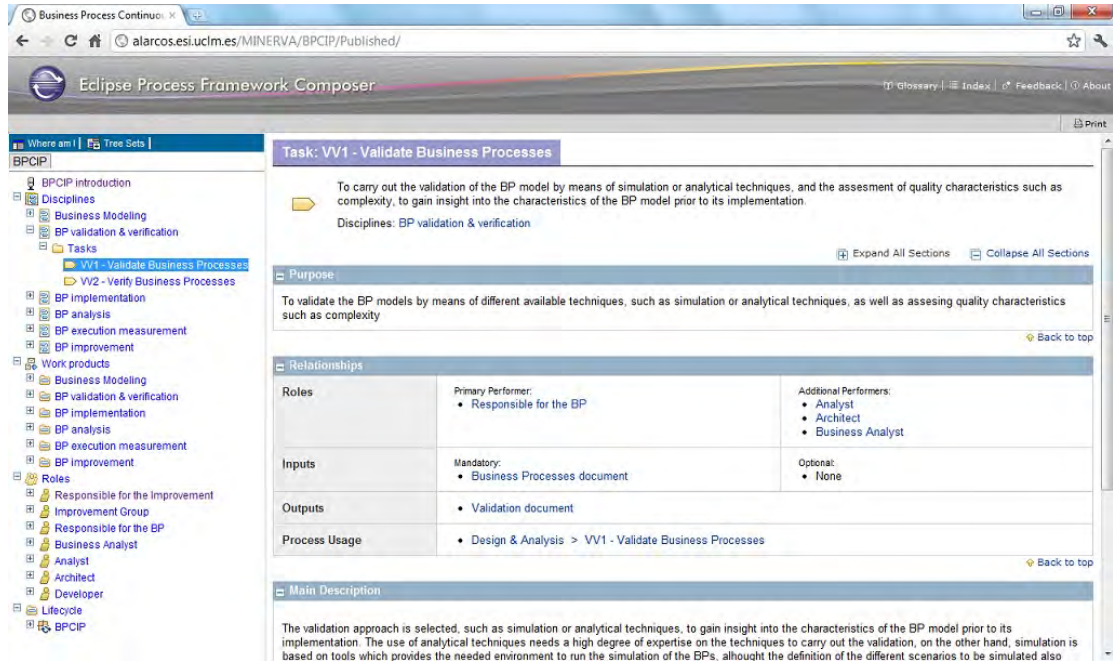


Figure B.3.: VV1-Validate Business Processes example

### B.2.1.3. BP Implementation Discipline

The BP Implementation Discipline presents the definition and description of the activities it comprises: I1-Implement BPs with services and I2-Reimplement services, from which the first one is presented in Figure B.4 as an example.

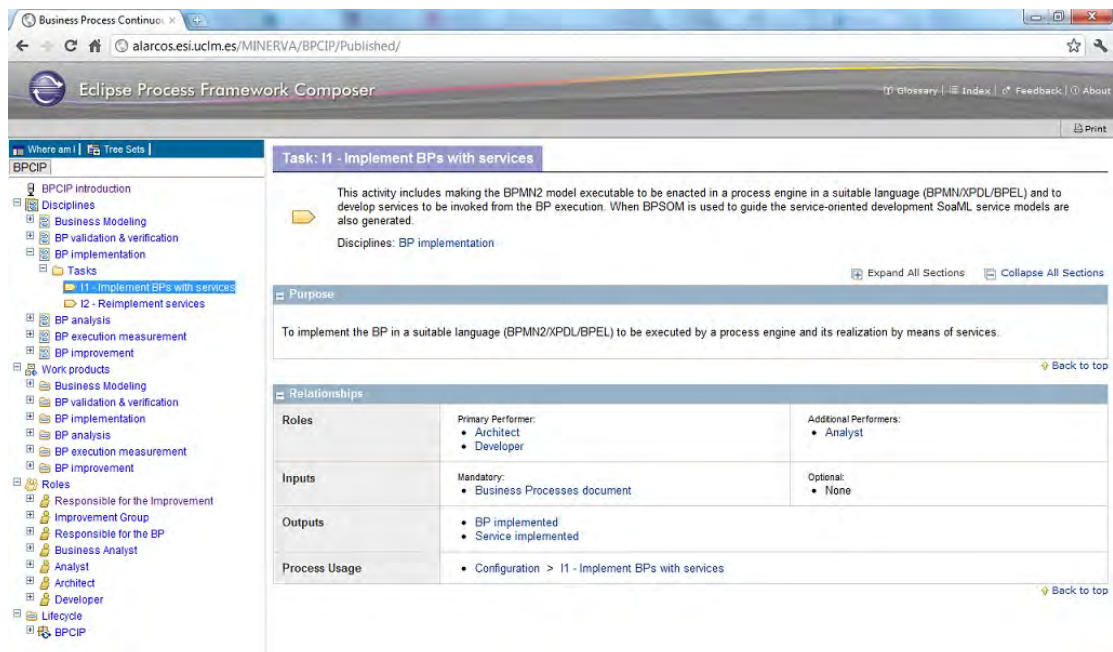


Figure B.4.: I1-Implement BPs with services example

### B.2.1.4. BP Analysis Discipline

The BP Analysis Discipline presents the definition and description of the activities it comprises: A1-Monitor BPs execution and A2-Analyze BPs execution, from which the first one is presented in Figure B.5 as an example.

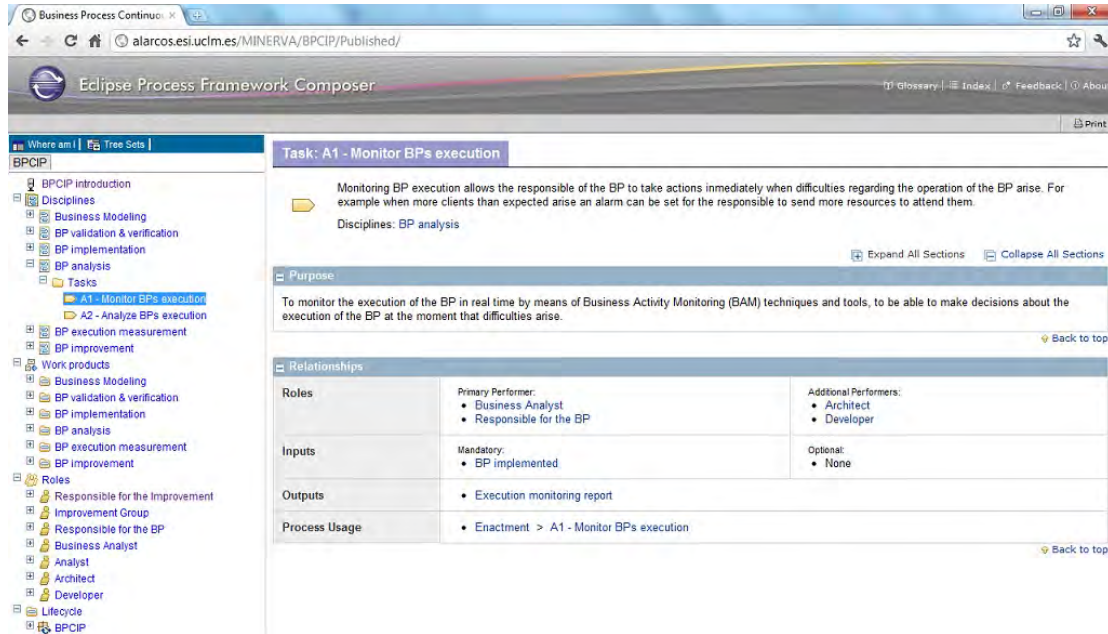


Figure B.5.: A1-Monitor BPs execution example

### B.2.1.5. BP Execution Measurement Discipline

The BP Execution Measurement Discipline presents the definition and description of the activities it comprises: EM1-Select execution measures, EM2-Implement execution measures collection, EM3-Collect execution measures and EM4-Analyze execution measurement results, from which the last one is presented in Figure B.6 as an example.

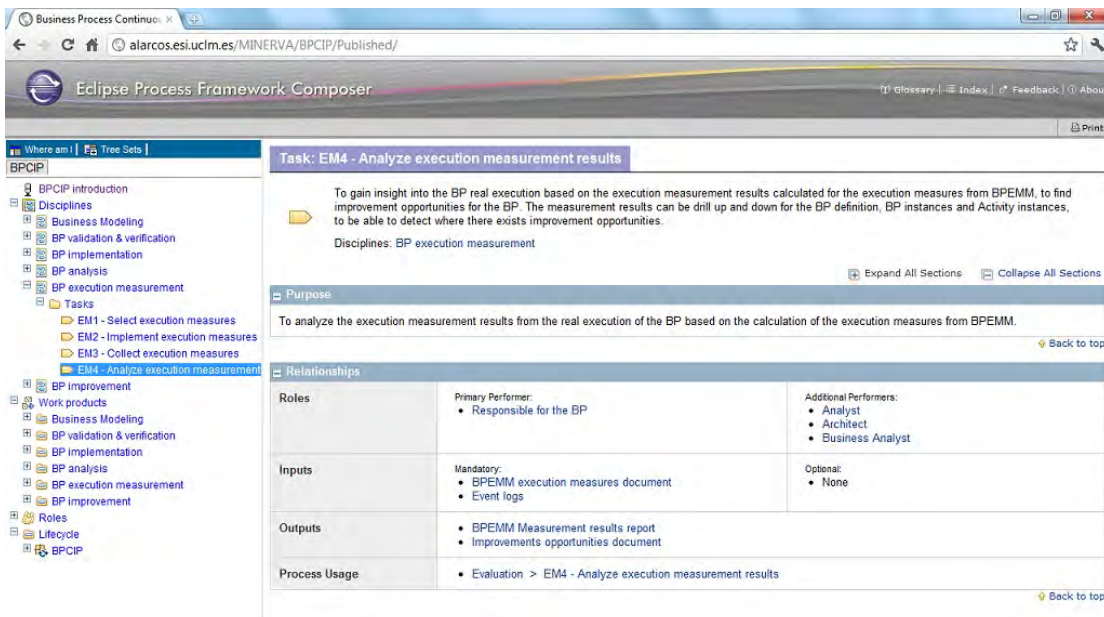


Figure B.6.: EM4-Analyze execution measurement results example

### B.2.1.6. BP Improvement Discipline

The BP Improvement Discipline presents the definition and description of the activities it comprises: IM1-Define improvements, EM2-Diagnose processes, EM3-Formulate improvements and EM4-Assess improvement effort, from which the first one is presented in Figure B.7 as an example.

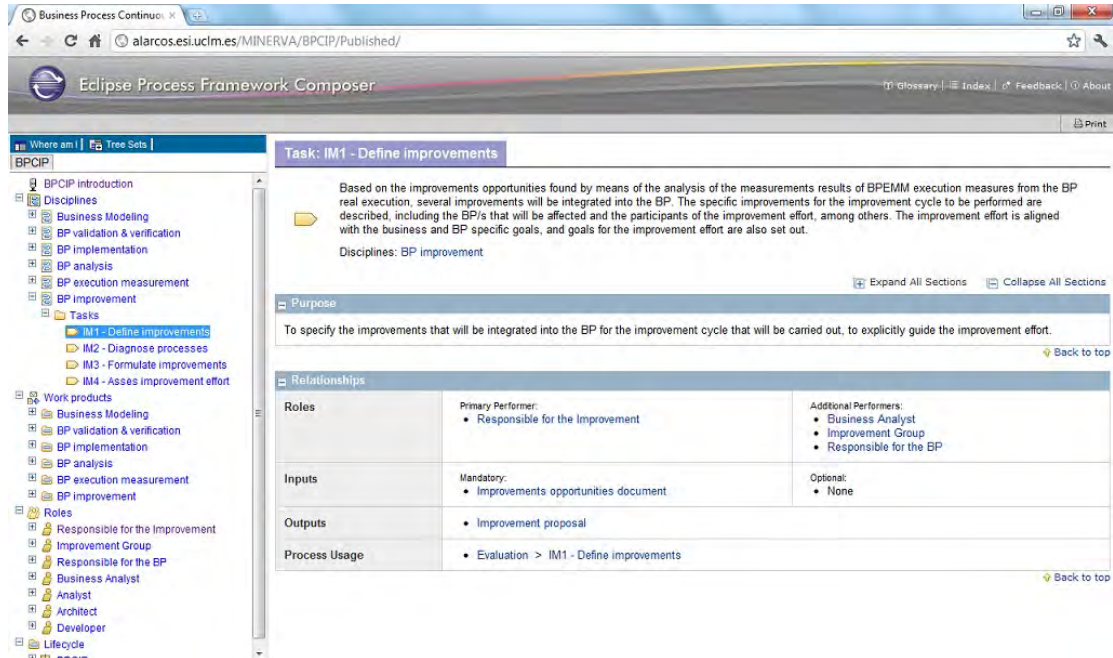


Figure B.7.: IM1-Define improvements example

### B.2.2. Work products

The work products folder contains the definition and description of artifacts used and generated by the defined activities. For each Discipline its work products are presented, from which the Event logs page is shown in Figure B.8 as an example. On the right side of the page a description of the artifact is presented, and a summary of the elements it comprises: roles responsible and modifying the artifact, the activities which use it as input and output, and its use in the defined lifecycle.

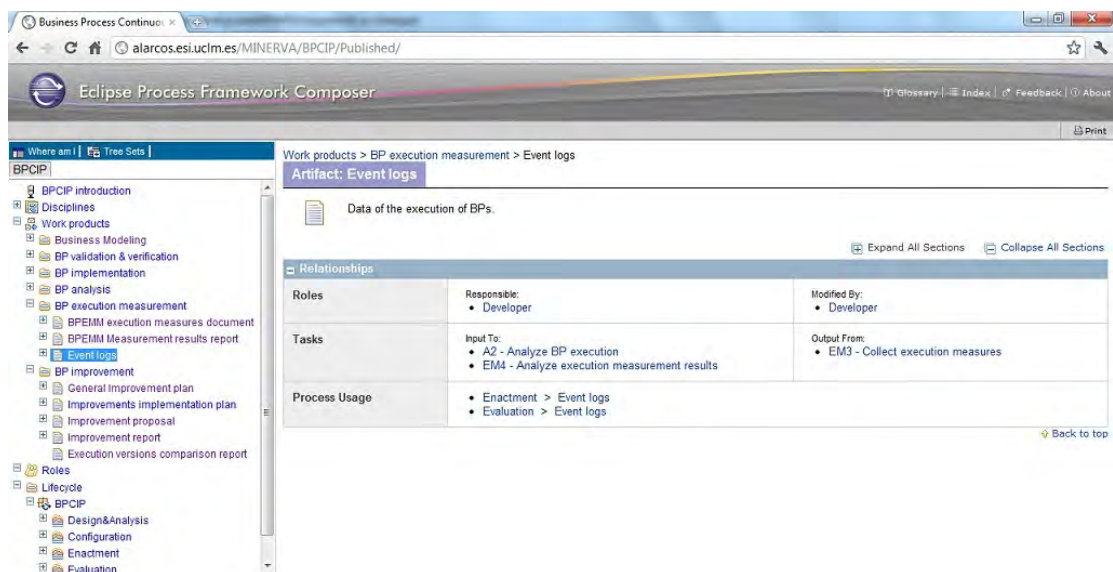


Figure B.8.: Event logs example

### B.2.3. Roles

The Roles folder contains the definition and description of the defined roles, showing in the first place, the activities it performs as primary performer and the artifacts it is responsible for, and then a summary of the activities it performs as additional performer (if any), the artifacts it modifies and its participation in the defined lifecycle. In Figure B.9 the page for the Responsible for the Improvement role description is shown as an example.

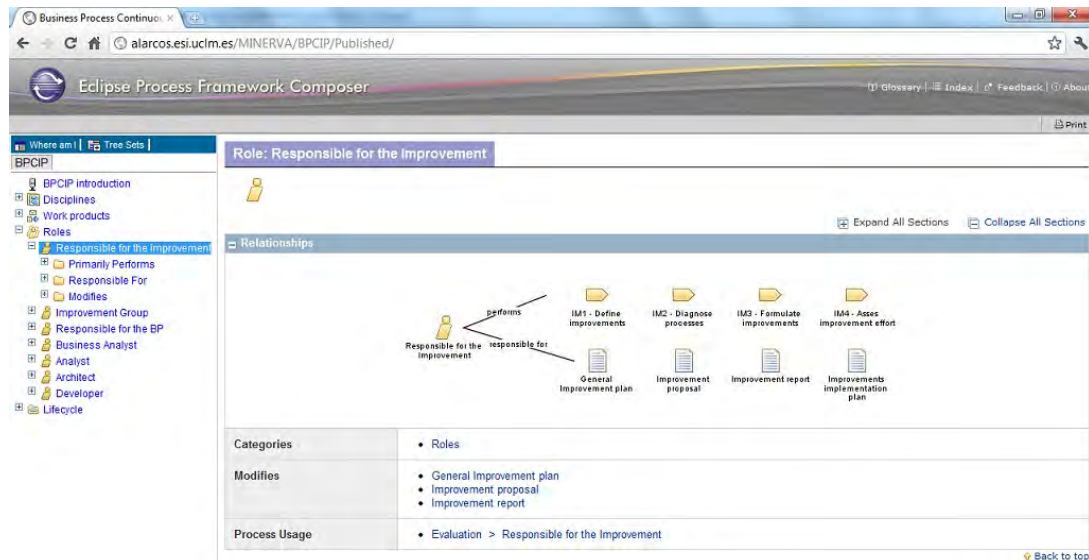


Figure B.9.: Responsible for the improvement role example

### B.2.4. Lifecycle

The Lifecycle folder contains the description of the defined lifecycle, in which the phases defined and the activities to be performed in each of them and their precedence, are presented. In Figure B.10 the page with the description of the Evaluation phase is presented as an example, showing on the left the activities to be performed, and on the right four tabs containing: the description of the phase, the work breakdown structure with an Activity Diagram of the realization of activities within the phase, the team allocation and the work products usage for the phase.

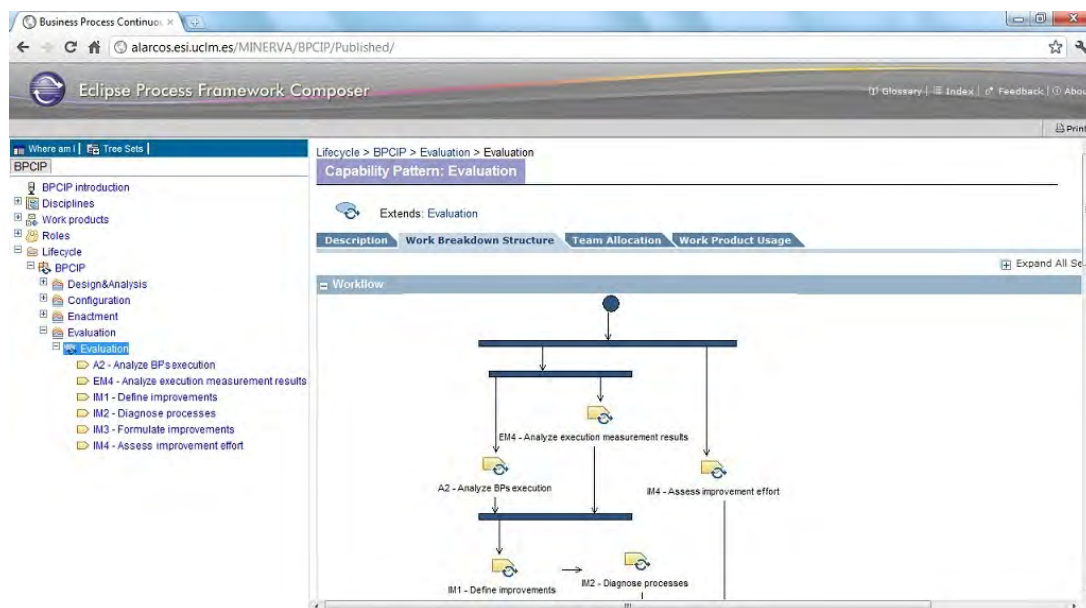


Figure B.10.: Evaluation phase example

## B.3. BPSOM Web Site

Once in the BPSOM Site the Introduction page is displayed, presenting a general description of BPSOM and its use as plug-in for other software development processes to guide the service-oriented development from BPs, as shown in Figure B.11. On the left side the structure of the site is shown as a tree, in the same way as presented for BPCIP in section B.2.

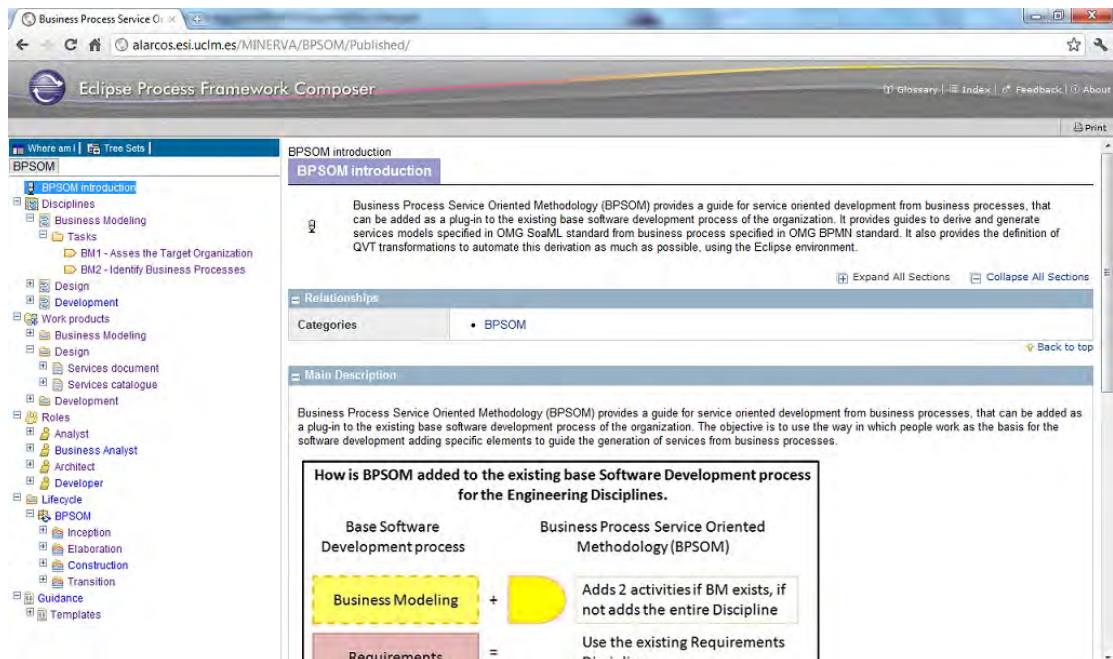


Figure B.11.: Introduction page in BPSOM Web Site from MINERVA

### B.3.1. Disciplines

As for BPCIP Disciplines, in the following a screenshot of an activity definition is presented as an example for each Discipline in BPCIP, where on the left side all the activities defined for the Discipline are displayed, and on the right side, the definition for the selected activity is shown, with the same elements as presented for BPCIP in subsection B.2.1.

### B.3.1.1. Business Modeling Discipline

The Business Modeling Discipline presents the description and elements for the two activities defined in it: BM1-Assess the target organization and BM2-Identify Business Processes. In Figure B.12 the page for the BM2-Identify Business Processes activity is shown as an example.

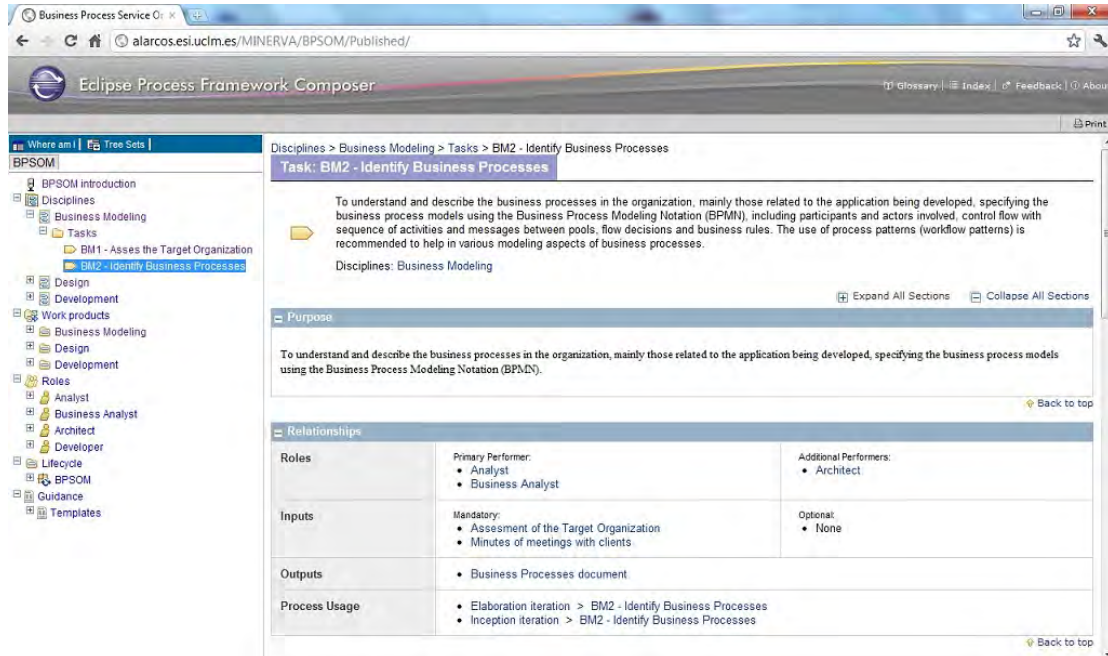


Figure B.12.: BM2-Identify Business Processes example

At the bottom of the page defined for this activity, an example of a “Grant Loan” BP modeled in BPMN2 is provided as shown in Figure B.13, which is then realized by means of the SoaML service model presented in the Design Discipline.

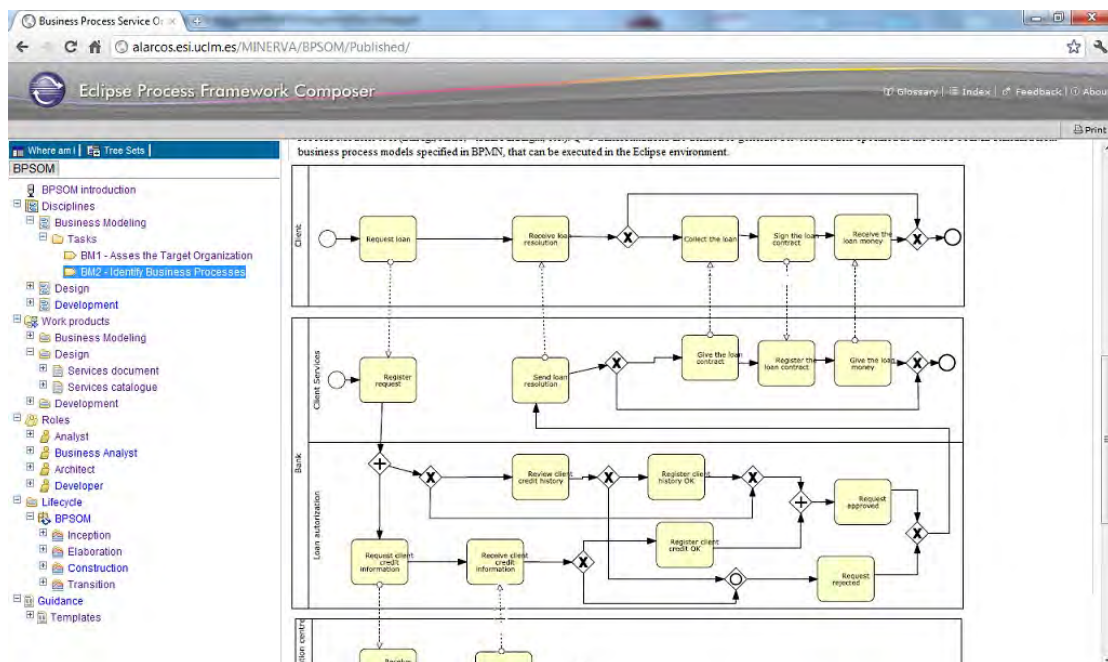


Figure B.13.: BM2-Identify Business Processes BPMN2 model example



### B.3.1.2. Design Discipline

The Design Discipline presents the description and elements for the five activities defined in it: D1-Identify and categorize services, D2-Specify services, D3-Investigate existing services, D4-Assign services to components and D5-Defined services interaction. For each activity an example is provided on which SoaML diagrams are to be generated to support the modeling of the realization of BPs by means of services, as presented in chapter 7. In Figure B.14 the page for the activity D1-Identify and categorize services is shown as an example.

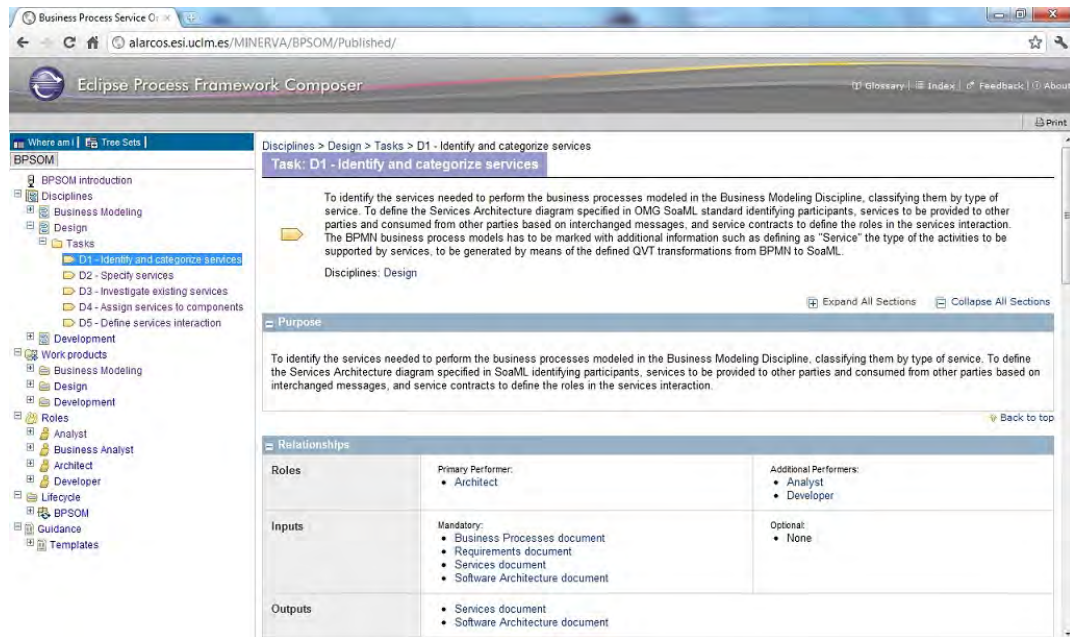


Figure B.14.: D1-Identify and categorize services example

At the bottom of the page defined for this activity, the corresponding SoaML diagrams for the “Grant Loan” BP for this activity are provided as shown in Figure B.15, as imported and visualized in the Eclipse SoaML plug-in.

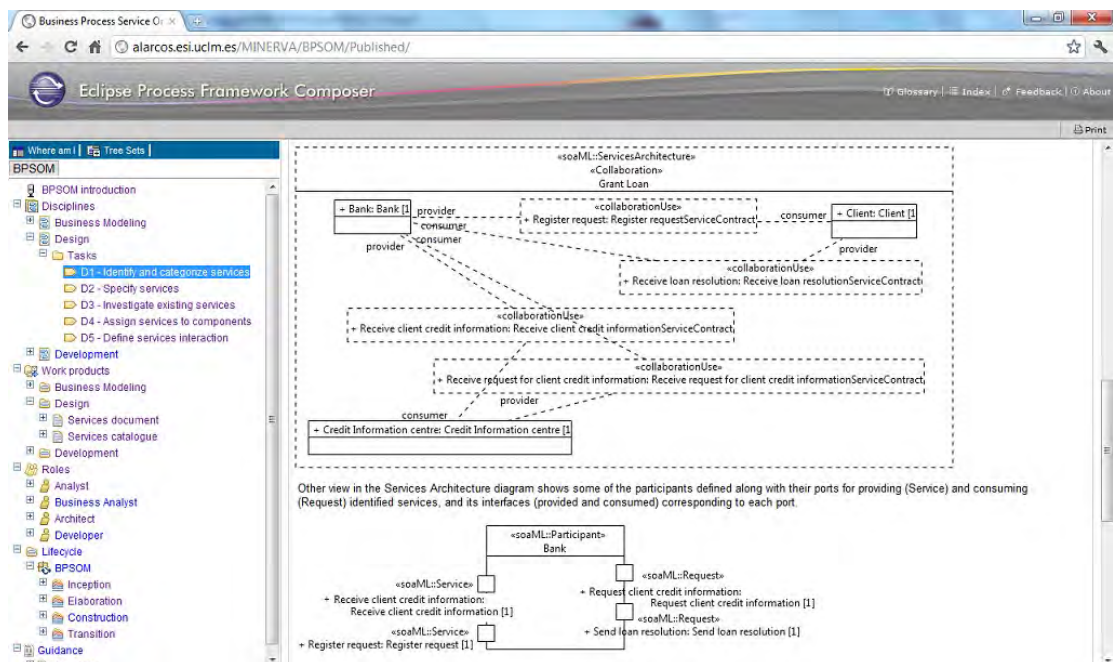


Figure B.15.: D1-Identify and categorize services SoaML diagrams

### B.3.1.3. Implementation Discipline

The Implementation Discipline presents the description and elements for the activity defined in it: I1-Implement services, whose page is shown in Figure B.16.

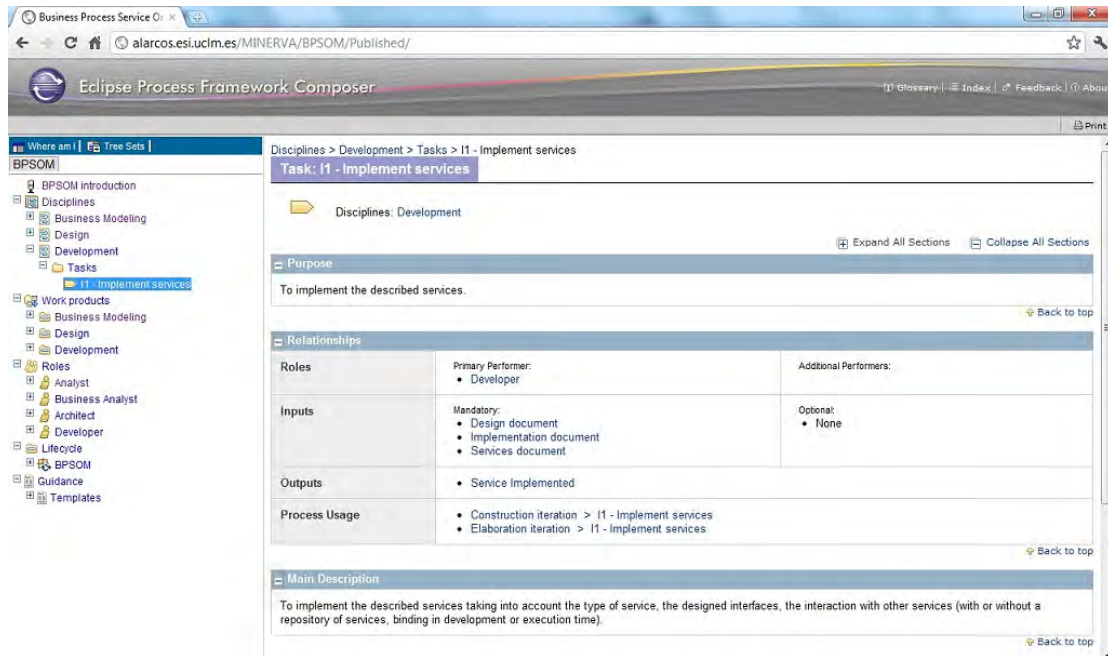


Figure B.16.: I1-Implement services

### B.3.2. Work products

The work products folder contains the definition and description of artifacts used and generated by the defined activities as presented from BPCIP in subsection B.2.2. For each Discipline its work products are presented, from which the Business Process document page is shown in Figure B.17 as an example.

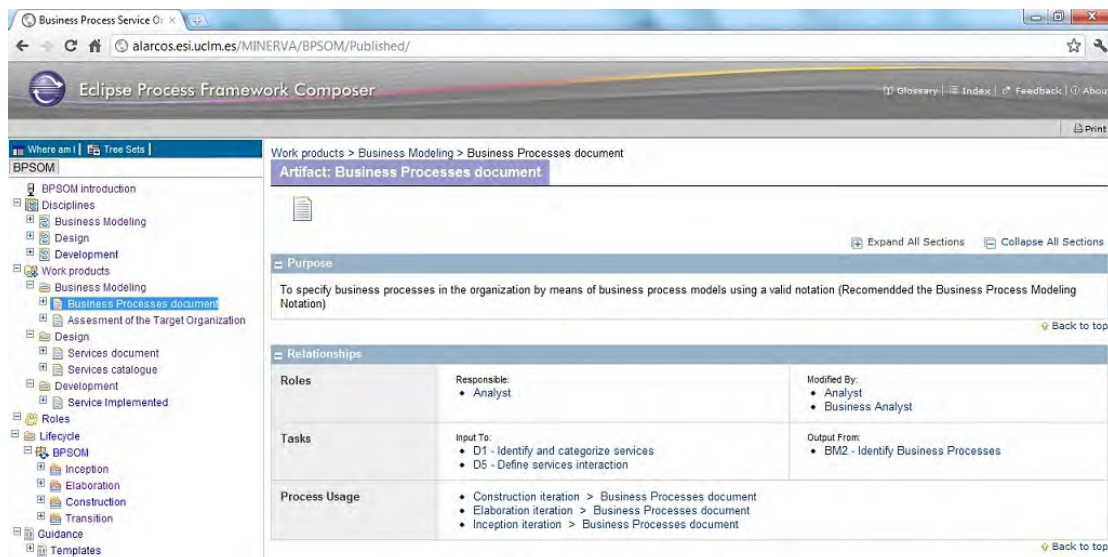


Figure B.17.: Work products defined in each Discipline

### B.3.3. Roles

The Roles folder contains the definition and description of the defined roles, showing the same elements as presented for BPCIP in subsection B.2.3. In Figure B.18 the page for the Architect role description is shown as an example.

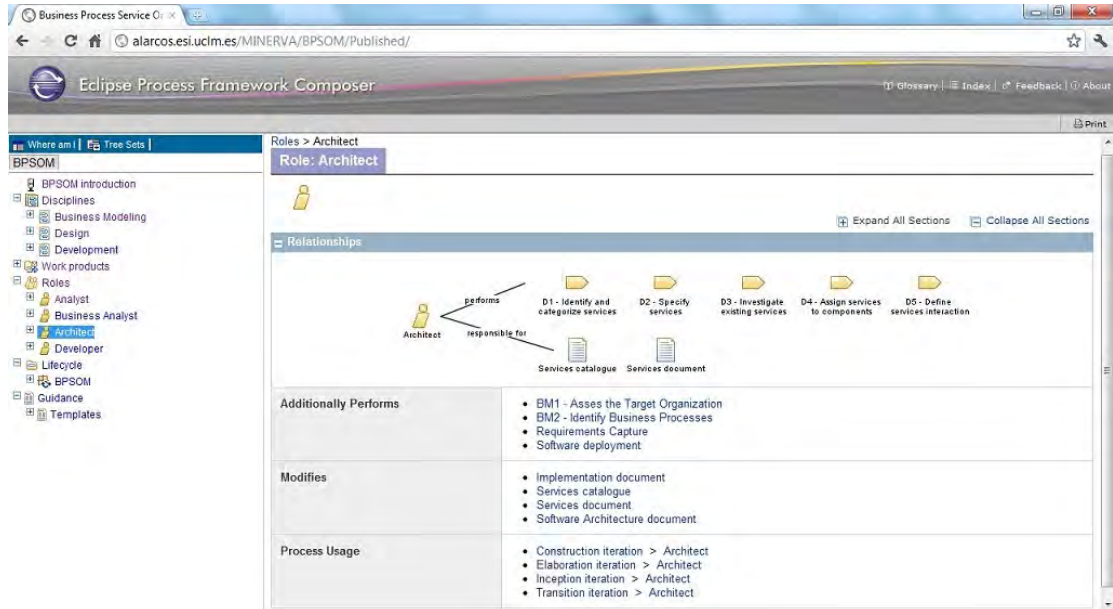


Figure B.18.: Architect role description example

### B.3.4. Lifecycle

The Lifecycle folder contains the definition and description of the defined lifecycle, in which suggested phases and activities to be performed in each of them and their precedence, are presented. In Figure B.19 the page with the description of the Elaboration phase is presented as an example, showing the same elements as presented for BPCIP in subsection B.2.4.

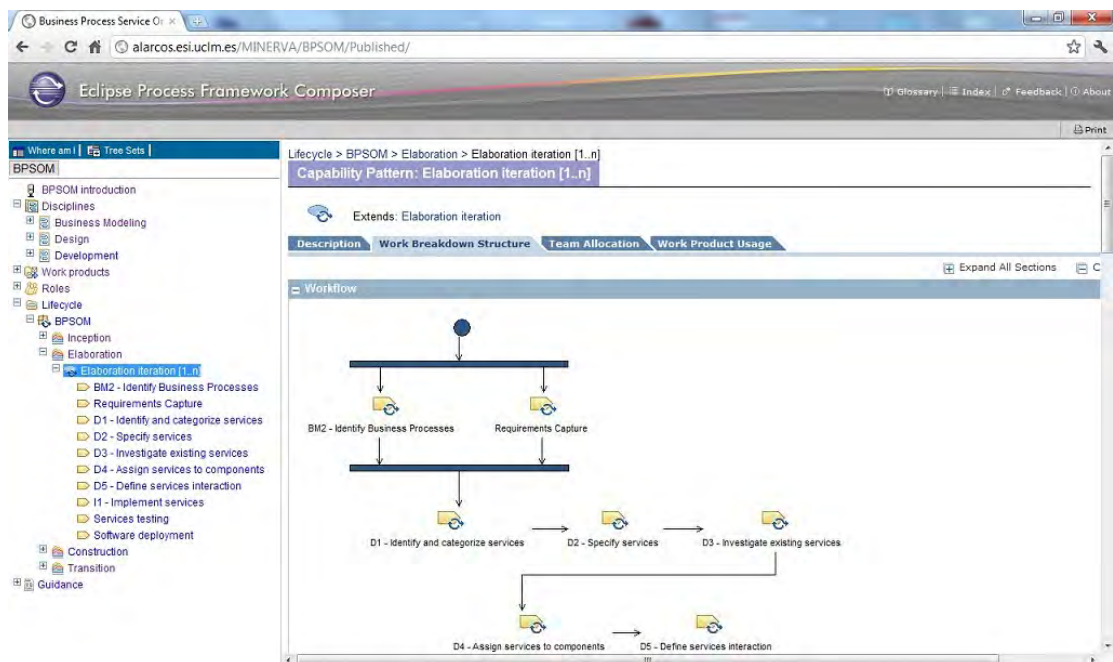


Figure B.19.: Elaboration phase description example



# Appendix C.

## QVT transformations code

### C.1. Overview

This Appendix presents the code for the QVT transformations as presented in chapter 8, for the three generation options for SoaML service models from ServiceTasks in the BPMN2 model, and for the three generation options adding interfaces, operations, parameters and messages in the BPMN2 model for the service provider.

### C.2. Generation from ServiceTask

In this section the code for the QVT transformations defined for the generation from ServiceTask identified in the BPMN2 model are presented: bidirectional option with simple UML Interfaces, bidirectional option with ServiceInterfaces and unidirectional option with simple UML Interfaces.

#### C.2.1. Bidirectional option with simple UML Interfaces

The bidirectional option with simple UML Interfaces generates one interface for the service provider and one interface for the service consumer, with the corresponding operation, parameters and messages for each one. The Service Port is typed with the interface of the provider and the Request Port with the one of the consumer. The Model, Participants, Messages, Services, ServiceContract and ServicesArchitecture are generated accordingly.

```
1 transformation bpmn2uml2soaml2 (bpmn:bpmn2, soaml:SoaML2) {
2   key uml::Interface {name};
3   key uml::Collaboration {name};
4   key uml::Class {name};
5   key uml::Package {name};
6   key uml::Property {name,type};
7   key uml::Operation {name};
8   key uml::Connector{end};
9   key uml::ConnectorEnd {role};
10  key uml::CollaborationUse{roleBinding};
11  key uml::Dependency {supplier,client};
12  key uml::Parameter {name};
13  key uml::Port {name};
14
15  /*top relation to define a SoaML model from the BPMN2 model of the BP*/
16  top relation ModelToSoaMLModel {
17    pn:String;
18    checkonly domain bpmn de:bpmn2::Definitions {name = pn};
19    enforce domain soaml mo:uml::Model{name = pn,
20      profileApplication = proa:uml::ProfileApplication{
```

```

21   appliedProfile = aproa:uml::Profile{
22     name = 'http://www.soaml.org/SoaML/1.0.0'}}};
23   enforce domain soaml sop:SoaML2::SoaMLPackage{
24     base_Package = mo};
25   }
26   /*top relation to define SoaML Participans from BPMN2 pools (process) of the BP*/
27   top relation ParticipantToSoaMLParticipant{
28     pn,pp:String;
29     sop:SoaML2::SoaMLPackage;
30     sa:SoaML2::Participant;
31     checkonly domain bpmn de:bpmn2::Definitions{name=pn,
32     rootElements=po:bpmn2::Process{name = pp}}};
33     enforce domain soaml mo:uml::Model{name = pn,
34     packagedElement = pa:uml::Package{name = 'Participants'+pn,
35     packagedElement = uma:uml::Class {name = pp}}}}};
36     enforce domain soaml sap:SoaML2::SoaMLPackage{
37     base_Package = pa};
38     when {
39       ModelToSoaMLModel(de,mo,sop);
40     }
41     where {
42       ParticipantsSoaMLrefs(po,uma,sa) or true;
43     }
44   }
45   relation ParticipantsSoaMLrefs{
46     pp:String;
47     checkonly domain bpmn po:bpmn2::Process{name = pp};
48     enforce domain soaml uma:uml::Class {name = pp};
49     enforce domain soaml sa:SoaML2::Participant{base_Class = uma};
50   }
51   /*top relation to define SoaML MessagesTypes from BPMN2 MessageFlows of the BP*/
52   top relation MessageToSoaMLMessage{
53     pn,cn,sn:String;
54     sop:SoaML2::SoaMLPackage;
55     se,se1:SoaML2::MessageType;
56
57     checkonly domain bpmn de:bpmn2::Definitions{name = pn,
58     rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
59     rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
60     rootElements=co:bpmn2::Collaboration{messageFlows=
61     me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}}}};
62     enforce domain soaml mo:uml::Model{name = pn,
63     packagedElement = pe:uml::Package{name = 'Messages'+pn},
64     packagedElement = ume:uml::Class {name = cn+'Message'},
65     packagedElement = ume1:uml::Class {name = sn+'Message'}}};
66     enforce domain soaml sep:SoaML2::SoaMLPackage{base_Package = pe};
67     when {
68       ModelToSoaMLModel(de,mo,sop);
69     }
70     where {
71       MessagesSoaMLrefs(po,p1,co,ume,ume1,se,se1) or true;
72     }
73   }
74   relation MessagesSoaMLrefs{
75     sn,cn:String;
76     checkonly domain bpmn po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}}};

```

```

77     checkonly domain bpmn p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}};
78     checkonly domain bpmn co:bpmn2::Collaboration
79     {messageFlows=me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}};
80     checkonly domain soaml ume:uml::Class {name = cn+'Message'};
81     checkonly domain soaml ume1:uml::Class {name = sn+'Message'};
82     enforce domain soaml se:SoaML2::MessageType{base_Class = ume};
83     enforce domain soaml se1:SoaML2::MessageType{base_Class = ume1};
84     }
85     /*top relation to define SoaML Services from BPMN2 ServiceTasks, Tasks and*/
86     /*MessageFlows of the BP*/
87     top relation ServicesToSoaMLServices{
88     pn,sn,cn:String;
89     pro:SoaML2::Provider;
90     cons:SoaML2::Consumer;
91     con:SoaML2::ServiceContract;
92     conn:SoaML2::ServiceChannel;
93     si:SoaML2::SoaMLPackage;
94     sop:SoaML2::SoaMLPackage;
95     checkonly domain bpmn de:bpmn2::Definitions{name = pn,
96     rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
97     rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
98     rootElements=co:bpmn2::Collaboration{messageFlows=
99     me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
100    enforce domain soaml mo:uml::Model{name = pn,
101    packagedElement = pi:uml::Package{name = 'Services'+pn,
102    packagedElement = se:uml::Package {name = cn+'Service'}}};
103    enforce domain soaml sip:SoaML2::SoaMLPackage{base_Package = pi};
104    when {
105    ModelToSoaMLModel(de,mo,sop);
106    }
107    where {
108        ElementsToSoaMLElements(st,sr,co,se,si,pro,cons,con,conn) or true;
109    }
110    }
111    relation ElementsToSoaMLElements{
112    cn,sn:String;
113    checkonly domain bpmn st:bpmn2::ServiceTask{name = cn};
114    checkonly domain bpmn sr:bpmn2::Task{name = sn};
115    checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
116    me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}};
117    enforce domain soaml se:uml::Package{name = cn+'Service',
118    packagedElement = it:uml::Interface{name= cn,
119    ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
120    visibility=uml::VisibilityKind::public,
121    ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1',
122    direction=uml::ParameterDirectionKind::inout},
123    ownedParameter = owpe:uml::Parameter{name=cn+'Return',
124    direction=uml::ParameterDirectionKind::return}}},
125    packagedElement = it1:uml::Interface{name = sn,
126    ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
127    visibility=uml::VisibilityKind::public,
128    ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
129    direction=uml::ParameterDirectionKind::inout},
130    ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
131    direction=uml::ParameterDirectionKind::return}}},
132    packagedElement = us:uml::Usage{supplier = it:uml::Interface{name=cn},

```

```

133 client = it1:uml::Interface{name=sn}},
134 packagedElement = us1:uml::Usage{supplier = it1:uml::Interface{name=sn},
135   client = it:uml::Interface{name=cn}},
136 packagedElement = col:uml::Collaboration{ name=cn+'ServiceContract',
137   ownedAttribute=ot:uml::Property{name='provider',type=it,
138   aggregation=uml::AggregationKind::composite},
139   ownedAttribute=ot1:uml::Property{name='consumer',type=it1,
140   aggregation=uml::AggregationKind::composite},
141   ownedConnector = oc:uml::Connector{
142     end = en:uml::ConnectorEnd {role = ot,isUnique=false},
143     end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}};
144 enforce domain soaml si:SoaML2::SoaMLPackage{base_Package = se};
145 enforce domain soaml pro:SoaML2::Provider{base_Interface = it};
146 enforce domain soaml cons:SoaML2::Consumer{base_Interface = it1};
147 enforce domain soaml con:SoaML2::ServiceContract{base_Collaboration = col};
148 enforce domain soaml conn:SoaML2::ServiceChannel {
149   base_Connector = oc:uml::Connector{}};
150 }
151 /*top relation to update SoaML Services adding MessageTypes as the*/
152 /*type of the parameters */
153 top relation MessageToServiceSoaMLMessage{
154   pn,cn,sn:String;
155   mes,mes1:SoaML2::MessageType;
156   si:SoaML2::SoaMLPackage;
157   pro:SoaML2::Provider;
158   cons:SoaML2::Consumer;
159   con:SoaML2::ServiceContract;
160   conn:SoaML2::ServiceChannel;
161   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
162   rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
163   rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
164   rootElements=co:bpmn2::Collaboration{messageFlows=
165   me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
166   checkonly domain soaml mo:uml::Model{name = pn,
167   packagedElement = pe : uml::Package{name = 'Messages'+pn,
168   packagedElement = mae:uml::Class {name = cn+'Message'},
169   packagedElement = mae1:uml::Class {name = sn+'Message'}},
170   packagedElement = pi : uml::Package {name = 'Services'+pn,
171     packagedElement = se :uml::Package{name = cn+'Service',
172     packagedElement = it:uml::Interface{name=cn,
173     ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
174     visibility=uml::VisibilityKind::public,
175     ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1',
176     direction=uml::ParameterDirectionKind::inout},
177     ownedParameter = owpe:uml::Parameter{name=cn+'Return',
178     direction=uml::ParameterDirectionKind::return}}},
179     packagedElement = it1:uml::Interface{name=sn,
180     ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
181     visibility=uml::VisibilityKind::public,
182     ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
183     direction=uml::ParameterDirectionKind::inout},
184     ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
185     direction=uml::ParameterDirectionKind::return}} } } };
186   enforce domain soaml mo:uml::Model{name = pn,
187   packagedElement = pi : uml::Package {name = 'Services'+pn,
188     packagedElement = se :uml::Package{name = cn+'Service',

```



```

189         packagedElement = it:uml::Interface{name=cn,
190 ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
191 visibility=uml::VisibilityKind::public,
192     ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1',
193     direction=uml::ParameterDirectionKind::inout, type = mae},
194     ownedParameter = owpe:uml::Parameter{name=cn+'Return',
195     direction=uml::ParameterDirectionKind::return, type = mae}}},
196
197         packagedElement = it1:uml::Interface{name=sn,
198 ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
199 visibility=uml::VisibilityKind::public,
200     ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
201     direction=uml::ParameterDirectionKind::inout, type = mae1},
202     ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
203     direction=uml::ParameterDirectionKind::return, type = mae1}}}}}};
204 when {
205     MessagesSoaMLrefs(po,p1,co,mae,mae1,mes,mes1);
206     ElementsToSoaMLElements(st,sr,co,se,si,pro,cons,con,conn);
207 }
208 }
209 /*top relation to update SoaML Participants adding Ports corresponding*/
210 /*to generated services */
211 top relation PortToSoaMLPort{
212 pn,an,cn,an1,sn:String;
213 spart,spart1:SoaML2::Participant;
214 ser:SoaML2::Service;
215 req:SoaML2::Request;
216 pro:SoaML2::Provider;
217 cons:SoaML2::Consumer;
218 con:SoaML2::ServiceContract;
219 conn:SoaML2::ServiceChannel;
220 si:SoaML2::SoaMLPackage;
221 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
222 rootElements=po:bpmn2::Process{name = an, flowElements=st:bpmn2::ServiceTask{name = cn}},
223 rootElements=p1:bpmn2::Process{name = an1, flowElements=sr:bpmn2::Task{name = sn}},
224 rootElements=co:bpmn2::Collaboration{messageFlows=
225 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
226 checkonly domain soaml mo:uml::Model{name = pn,
227 packagedElement = pa : uml::Package{name = 'Participants'+pn,
228 packagedElement = pae:uml::Class {name = an},
229 packagedElement = pae1:uml::Class {name = an1}},
230     packagedElement = pi : uml::Package {name = 'Services'+pn,
231     packagedElement = se :uml::Package{name = cn+'Service',
232     packagedElement = it:uml::Interface{name=cn},
233     packagedElement = it1:uml::Interface{name=sn}}}}};
234 enforce domain soaml mo:uml::Model{name = pn,
235 packagedElement = pa : uml::Package{name = 'Participants'+pn,
236 packagedElement = pae:uml::Class {name = an,
237     ownedAttribute = opor:uml::Port{ name = cn, isService=true, type=it}},
238 packagedElement = pae1:uml::Class {name = an1,
239     ownedAttribute = opor1:uml::Port{name = sn, isService=true, type=it1}}}}};
240 when {
241     ParticipantsSoaMLrefs(po,pae,spart);
242     ParticipantsSoaMLrefs(p1,pae1,spart1);
243     ElementsToSoaMLElements(st,sr,co,se,si,pro,cons,con,conn);
244 }

```

```

245 where {
246 PortSoaMLrefs(po,p1,co,paе,paе1,ser,req) or true;
247 }
248 }
249 relation PortSoaMLrefs {
250 an,cn,an1,sn:String;
251 checkonly domain bpmn po:bpmn2::Process{name = an,
252     flowElements=st:bpmn2::ServiceTask{name = cn}};
253 checkonly domain bpmn p1:bpmn2::Process{name = an1,
254     flowElements=sr:bpmn2::Task {name = sn}};
255     checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
256 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}};
257 enforce domain soaml paе:uml::Class {name = an,
258     ownedAttribute = opor:uml::Port{name = cn} };
259 enforce domain soaml paе1:uml::Class {name = an1,
260     ownedAttribute = opor1:uml::Port{name = sn} };
261 enforce domain soaml ser:SoaML2::Service{base_Port = opor};
262 enforce domain soaml req:SoaML2::Request{base_Port = opor1};
263 }
264 /*top relation to define SoaML ServicesArchitecture from BPMN2 Collaboration*/
265 /*and generated SoaML elements */
266 top relation CollaborationToSoaMLServicesArchitecture{
267 pn,sn:String;
268 sop:SoaML2::SoaMLPackage;
269 su:SoaML2::ServicesArchitecture;
270 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
271     rootElements=co:bpmn2::Collaboration{name = sn} };
272 enforce domain soaml mo:uml::Model{name = pn,
273     packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
274     packagedElement = uma:uml::Collaboration {name = sn}}};
275     enforce domain soaml sup:SoaML2::SoaMLPackage{base_Package = pu};
276 when {
277 ModelToSoaMLModel(de,mo,sop);
278 }
279 where {
280 ServicesArchElementsToSoaMLServicesArchElements(de,mo,mo,su)or true;
281 }
282 }
283 relation ServicesArchElementsToSoaMLServicesArchElements{
284 pn,sn,bn,cn,an,an1:String;
285 pro:SoaML2::Provider;
286 cons:SoaML2::Consumer;
287 con:SoaML2::ServiceContract;
288 conn:SoaML2::ServiceChannel;
289 si:SoaML2::SoaMLPackage;
290 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
291     rootElements=po:bpmn2::Process{name = an,
292     flowElements=st:bpmn2::ServiceTask{name = cn}},
293     rootElements=p1:bpmn2::Process{name = an1,
294     flowElements=sr:bpmn2::Task {name = sn}},
295     rootElements=co:bpmn2::Collaboration{name = bn,
296     messageFlows = msf:bpmn2::MessageFlow{targetRef = st, sourceRef = sr},
297     participants = par:bpmn2::Participant{name = an, processRef = po},
298     participants = par1:bpmn2::Participant {name = an1, processRef = p1}}};
299 checkonly domain soaml mo:uml::Model{name = pn,
300     packagedElement = pa : uml::Package{name = 'Participants'+pn,

```

```

301 packagedElement = pae:uml::Class {name = an},
302 packagedElement = pae1:uml::Class {name = an1}},
303   packagedElement = pi : uml::Package {name = 'Services'+pn,
304     packagedElement = se :uml::Package{name = cn+'Service',
305       packagedElement = it:uml::Interface{name=cn},
306       packagedElement = it1:uml::Interface{name=sn},
307       packagedElement = col:uml::Collaboration{name=cn+'ServiceContract',
308         ownedAttribute=ot:uml::Property{name='provider',type=it,
309           aggregation=uml::AggregationKind::composite},
310         ownedAttribute=ot1:uml::Property{name='consumer',type=it1,
311           aggregation=uml::AggregationKind::composite}}}}};
312 enforce domain soaml mo:uml::Model{name = pn,
313   packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
314     packagedElement = uma:uml::Collaboration {name = bn,
315       ownedAttribute = oa:uml::Property{name = an, type = pae,
316         aggregation = uml::AggregationKind::composite},
317       ownedAttribute = oa1:uml::Property{name = an1, type = pae1,
318         aggregation = uml::AggregationKind::composite},
319
320       collaborationUse = cu:uml::CollaborationUse {name = cn, type = col,
321         roleBinding = rb:uml::Dependency{supplier = oa:uml::Property{},
322         client = ot:uml::Property{},client = cu:uml::CollaborationUse{}},
323         roleBinding = rb1:uml::Dependency{supplier = oa1:uml::Property{type = pae1},
324         client = ot1:uml::Property{}, client = cu:uml::CollaborationUse{type = col}}}}}}};
325 enforce domain soaml su:SoaML2::ServicesArchitecture{base_Collaboration = uma};
326 when {
327     ElementsToSoamLElements(st,sr,co,se,si,pro,cons,con,conn);
328 }
329 }
330 }

```

### C.2.2. Bidirectional option with ServiceInterfaces

The bidirectional option with ServiceInterfaces also generates one interface for the service provider and one interface for the service consumer, with the corresponding operation, parameters and messages for each one, and a ServiceInterface that realizes the interface of the provider and uses the interface of the consumer. Both the Service Port and the Request Port are typed with the ServiceInterface, the consumer with conjugated option. The Model, Participants, Messages, Services, ServiceContract and ServicesArchitecture are generated accordingly.

```

1 transformation bpmn2uml2soaml2 (bpmn:bpmn2, soaml:SoaML2) {
2   key uml::Interface {name};
3   key uml::Collaboration {name};
4   key uml::Class {name};
5   key uml::Package {name};
6   key uml::Property {name,type};
7   key uml::Operation {name};
8   key uml::Connector{end};
9   key uml::ConnectorEnd {role};
10  key uml::CollaborationUse{roleBinding};
11  key uml::Dependency {supplier,client};
12  key uml::Parameter {name};
13  key uml::Port{name};
14
15  /*top relation to define a SoaML model from the BPMN2 model of the BP*/

```

```

16 top relation ModelToSoaMLModel {
17   pn:String;
18   checkonly domain bpmn de:bpmn2::Definitions {name = pn};
19   enforce domain soaml mo:uml::Model{name = pn,
20   profileApplication = proa:uml::ProfileApplication{
21   appliedProfile = aproa:uml::Profile{
22   name = 'http://www.soaml.org/SoaML/1.0.0'}}};
23   enforce domain soaml sop:SoaML2::SoaMLPackage{
24   base_Package = mo};
25 }
26 /*top relation to define SoaML Participans from BPMN2 pools (process) of the BP*/
27 top relation ParticipantToSoaMLParticipant{
28   pn,pp:String;
29   sop:SoaML2::SoaMLPackage;
30   sa:SoaML2::Participant;
31   checkonly domain bpmn de:bpmn2::Definitions{name=pn,
32   rootElements=po:bpmn2::Process{name = pp}};
33   enforce domain soaml mo:uml::Model{name = pn,
34   packagedElement = pa:uml::Package{name = 'Participants'+pn,
35   packagedElement = uma:uml::Class {name = pp}}};
36   enforce domain soaml sap:SoaML2::SoaMLPackage{
37   base_Package = pa};
38   when {
39     ModelToSoaMLModel(de,mo,sop);
40   }
41   where {
42     ParticipantsSoaMLrefs(po,uma,sa) or true;
43   }
44 }
45 relation ParticipantsSoaMLrefs{
46   pp:String;
47   checkonly domain bpmn po:bpmn2::Process{name = pp};
48   enforce domain soaml uma:uml::Class {name = pp};
49   enforce domain soaml sa:SoaML2::Participant{base_Class = uma};
50 }
51 /*top relation to define SoaML MessagesTypes from BPMN2 MessageFlows of the BP*/
52 top relation MessageToSoaMLMessage{
53   pn,cn,sn:String;
54   sop:SoaML2::SoaMLPackage;
55   se,se1:SoaML2::MessageType;
56   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
57   rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
58   rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
59   rootElements=co:bpmn2::Collaboration{messageFlows=
60   me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
61   enforce domain soaml mo:uml::Model{name = pn,
62   packagedElement = pe:uml::Package{name = 'Messages'+pn},
63   packagedElement = ume:uml::Class {name = cn+'Message'},
64   packagedElement = ume1:uml::Class {name = sn+'Message'}};
65   enforce domain soaml sep:SoaML2::SoaMLPackage{base_Package = pe};
66   when {
67     ModelToSoaMLModel(de,mo,sop);
68   }
69   where {
70     MessagesSoaMLrefs(po,p1,co,ume,ume1,se,se1) or true;
71   }

```

```

72 }
73 relation MessagesSoaMLrefs{
74   sn,cn:String;
75   checkonly domain bpmn po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}};
76   checkonly domain bpmn p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}};
77   checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
78   me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}};
79   checkonly domain soaml ume:uml::Class {name = cn+'Message'};
80   checkonly domain soaml ume1:uml::Class {name = sn+'Message'};
81   enforce domain soaml se:SoaML2::MessageType{base_Class = ume};
82   enforce domain soaml se1:SoaML2::MessageType{base_Class = ume1};
83 }
84 /*top relation to define SoaML Services from BPMN2 ServiceTasks, Tasks */
85 /*and MessageFlows of the BP*/
86 top relation ServicesToSoaMLServices{
87   pn,sn,cn:String;
88   sint:SoaML2::ServiceInterface;
89   pro:SoaML2::Provider;
90   cons:SoaML2::Consumer;
91   con:SoaML2::ServiceContract;
92   conn:SoaML2::ServiceChannel;
93   si:SoaML2::SoaMLPackage;
94   sop:SoaML2::SoaMLPackage;
95   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
96   rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
97   rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
98   rootElements=co:bpmn2::Collaboration{messageFlows=
99   me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
100   enforce domain soaml mo:uml::Model{name = pn,
101   packagedElement = pi:uml::Package{name = 'Services'+pn,
102   packagedElement = se:uml::Package {name = cn+'Service'}}};
103   enforce domain soaml sip:SoaML2::SoaMLPackage{base_Package = pi};
104   when {
105   ModelToSoaMLModel(de,mo,sop);
106   }
107   where {
108     ElementsToSoaMLElements(st,sr,co,se,si,sint,pro,cons,con,conn) or true;
109   }
110 }
111 relation ElementsToSoaMLElements{
112   cn,sn:String;
113   checkonly domain bpmn st:bpmn2::ServiceTask{name = cn };
114   checkonly domain bpmn sr:bpmn2::Task{name = sn };
115   checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
116   me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}};
117   enforce domain soaml se:uml::Package{name = cn+'Service',
118   packagedElement = sin:uml::Class{name=cn,
119   clientDependency = cd:uml::Dependency{
120   supplier = us:uml::Usage{}, supplier=ir:uml::InterfaceRealization{}},
121   interfaceRealization=ir:uml::InterfaceRealization{
122   supplier=it:uml::Interface{}},contract=it:uml::Interface{}},
123   client=sin:uml::Class{}}},
124   packagedElement = us:uml::Usage{supplier = it1:uml::Interface{}},
125   client = sin:uml::Class{}},
126   packagedElement = it:uml::Interface{name= cn,
127   ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',

```

```

128 visibility=uml::VisibilityKind::public,
129 ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1',
130 direction=uml::ParameterDirectionKind::inout},
131 ownedParameter = owpe:uml::Parameter{name=cn+'Return',
132 direction=uml::ParameterDirectionKind::return}}},
133 packagedElement = it1:uml::Interface{name = sn,
134     ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
135     visibility=uml::VisibilityKind::public,
136     ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
137     direction=uml::ParameterDirectionKind::inout},
138     ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
139     direction=uml::ParameterDirectionKind::return}}},
140     packagedElement = col:uml::Collaboration{ name=cn,
141     ownedAttribute=ot:uml::Property{name='provider',type=sin,
142     aggregation=uml::AggregationKind::composite},
143     ownedAttribute=ot1:uml::Property{name='consumer', type=sin,
144     aggregation=uml::AggregationKind::composite},
145     ownedConnector = oc:uml::Connector{
146     end = en:uml::ConnectorEnd {role = ot,isUnique=false},
147     end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}}};
148 enforce domain soaml si:SoaML2::SoaMLPackage{base_Package = se};
149 enforce domain soaml sint:SoaML2::ServiceInterface{ base_Class = sin};
150 enforce domain soaml pro:SoaML2::Provider{base_Interface = it};
151 enforce domain soaml cons:SoaML2::Consumer{ base_Interface = it1};
152 enforce domain soaml con:SoaML2::ServiceContract{base_Collaboration = col};
153 enforce domain soaml conn:SoaML2::ServiceChannel {base_Connector = oc:uml::Connector{}};
154 }
155 /*top relation to update SoaML Services adding MessageTypes as the*/
156 /*type of the parameters */
157 top relation MessageToServiceSoaMLMessage{
158 pn,cn,sn:String;
159 mes,mes1:SoaML2::MessageType;
160 si:SoaML2::SoaMLPackage;
161 sint:SoaML2::ServiceInterface;
162 pro:SoaML2::Provider;
163 cons:SoaML2::Consumer;
164 con:SoaML2::ServiceContract;
165 conn:SoaML2::ServiceChannel;
166 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
167 rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
168 rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
169 rootElements=co:bpmn2::Collaboration{messageFlows=
170 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
171 checkonly domain soaml mo:uml::Model{name = pn,
172 packagedElement = pe : uml::Package{name = 'Messages'+pn,
173 packagedElement = mae:uml::Class {name = cn+'Message'},
174 packagedElement = mae1:uml::Class {name = sn+'Message'}},
175     packagedElement = pi : uml::Package {name = 'Services'+pn,
176     packagedElement = se :uml::Package{name = cn+'Service',
177 packagedElement = it:uml::Interface{name=cn,
178     ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
179     visibility=uml::VisibilityKind::public,
180     ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1',
181     direction=uml::ParameterDirectionKind::inout},
182     ownedParameter = owpe:uml::Parameter{name=cn+'Return',
183     direction=uml::ParameterDirectionKind::return}}},

```

```

184 packagedElement = it1:uml::Interface{name=sn,
185 ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
186 visibility=uml::VisibilityKind::public,
187 ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
188 direction=uml::ParameterDirectionKind::inout},
189 ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
190 direction=uml::ParameterDirectionKind::return}}} } };
191 enforce domain soaml mo:uml::Model{name = pn,
192   packagedElement = pi : uml::Package {name = 'Services'+pn,
193     packagedElement = se :uml::Package{name = cn+'Service',
194 packagedElement = it:uml::Interface{name=cn,
195   ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
196   visibility=uml::VisibilityKind::public,
197   ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1',
198 direction=uml::ParameterDirectionKind::inout, type = mae},
199   ownedParameter = owpe:uml::Parameter{name=cn+'Return',
200   direction=uml::ParameterDirectionKind::return, type = mae}}}},
201 packagedElement = it1:uml::Interface{name=sn,
202 ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
203   visibility=uml::VisibilityKind::public,
204 ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
205 direction=uml::ParameterDirectionKind::inout, type = mae1},
206   ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
207   direction=uml::ParameterDirectionKind::return, type = mae1}}}}}}};
208 when {
209   MessagesSoAMLrefs(po,p1,co,mae,mae1,mes,mes1);
210   ElementsToSoAMLElements(st,sr,co,se,si,sint,pro,cons,con,conn);
211 }
212 }
213 /*top relation to update SoaML Participants adding Ports corresponding*/
214 /*to generated services */
215 top relation PortToSoaMLPort{
216 pn,an,cn,an1,sn:String;
217 spart,spart1:SoaML2::Participant;
218 ser:SoaML2::Service;
219 req:SoaML2::Request;
220 sint:SoaML2::ServiceInterface;
221 pro:SoaML2::Provider;
222 cons:SoaML2::Consumer;
223 con:SoaML2::ServiceContract;
224 conn:SoaML2::ServiceChannel;
225 si:SoaML2::SoaMLPackage;
226 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
227 rootElements=po:bpmn2::Process{name = an,
228 flowElements=st:bpmn2::ServiceTask{name = cn}},
229 rootElements=p1:bpmn2::Process{name = an1,
230 flowElements=sr:bpmn2::Task{name = sn}},
231 rootElements=co:bpmn2::Collaboration{messageFlows=
232 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
233 checkonly domain soaml mo:uml::Model{name = pn,
234 packagedElement = pa : uml::Package{name = 'Participants'+pn,
235 packagedElement = pae:uml::Class {name = an},
236 packagedElement = pae1:uml::Class {name = an1}},
237   packagedElement = pi : uml::Package {name = 'Services'+pn,
238     packagedElement = se :uml::Package{name = cn+'Service',
239 packagedElement = sin:uml::Class{name=cn}}}}};

```

```

240 enforce domain soaml mo:uml::Model{name = pn,
241 packagedElement = pa : uml::Package{name = 'Participants'+pn,
242 packagedElement = pae:uml::Class {name = an,
243   ownedAttribute = opor:uml::Port{ name = cn,
244     isService=true, type=sin}},
245 packagedElement = pae1:uml::Class {name = an1,
246   ownedAttribute = opor1:uml::Port{name = sn,
247     isService=true, type=sin,isConjugated=true}}}}};
248 when {
249   ParticipantsSoaMLrefs(po,pae,spart);
250   ParticipantsSoaMLrefs(p1,pael,spart1);
251   ElementsToSoaMLElements(st,sr,co,se,si,sint,pro,cons,con,conn);
252 }
253 where {
254 PortSoaMLrefs(po,p1,co,pael,pael,ser,req) or true;
255 }
256 }
257 relation PortSoaMLrefs {
258 an,cn,an1,sn:String;
259 checkonly domain bpmn po:bpmn2::Process{name = an,
260   flowElements=st:bpmn2::ServiceTask{name = cn}}};
261 checkonly domain bpmn p1:bpmn2::Process{name = an1,
262   flowElements=sr:bpmn2::Task {name = sn}}};
263   checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
264 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
265 enforce domain soaml pae:uml::Class {name = an,
266   ownedAttribute = opor:uml::Port{name = cn}}};
267 enforce domain soaml pae1:uml::Class {name = an1,
268   ownedAttribute = opor1:uml::Port{name = sn}}};
269 enforce domain soaml ser:SoaML2::Service{base_Port = opor};
270 enforce domain soaml req:SoaML2::Request{base_Port = opor1};
271 }
272 /*top relation to define SoaML ServicesArchitecture from BPMN2 Collaboration*/
273 /*and generated SoaML elements */
274 top relation CollaborationToSoaMLServicesArchitecture{
275 pn,an:String;
276 sop:SoaML2::SoaMLPackage;
277 su:SoaML2::ServicesArchitecture;
278 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
279 rootElements=co:bpmn2::Collaboration{name = an}
280 };
281 enforce domain soaml mo:uml::Model{name = pn,
282 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
283 packagedElement = uma:uml::Collaboration {name = an}}};
284   enforce domain soaml sup:SoaML2::SoaMLPackage{base_Package = pu };
285 when {
286 ModelToSoaMLModel(de,mo,sop);
287 }
288 where {
289 ServicesArchElementsToSoaMLServicesArchElements(de,mo,mo,su) or true;
290 }
291 }
292 relation ServicesArchElementsToSoaMLServicesArchElements{
293 pn,sn,bn,cn,an,an1:String;
294 sint:SoaML2::ServiceInterface;
295 pro:SoaML2::Provider;

```



```

296 cons:SoaML2::Consumer;
297 con:SoaML2::ServiceContract;
298 conn:SoaML2::ServiceChannel;
299 si:SoaML2::SoaMLPackage;
300 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
301 rootElements=co:bpmn2::Collaboration{name = bn,
302   participants = par:bpmn2::Participant{name = an,
303     processRef = po:bpmn2::Process{name = an}},
304   participants = par1:bpmn2::Participant {name = an1,
305     processRef = p1:bpmn2::Process{name = an1}},
306   messageFlows = msf:bpmn2::MessageFlow{
307     targetRef = st:bpmn2::ServiceTask{name = cn},
308     sourceRef = sr:bpmn2::Task {name = sn}}},
309     rootElements=po:bpmn2::Process{name = an,
310       flowElements=st:bpmn2::ServiceTask{name = cn}},
311     rootElements=p1:bpmn2::Process{name = an1,
312       flowElements=sr:bpmn2::Task {name = sn}}}};
313 checkonly domain soaml mo:uml::Model{name = pn,
314 packagedElement = pa : uml::Package{name = 'Participants'+pn,
315 packagedElement = pae:uml::Class {name = an},
316 packagedElement = pae1:uml::Class {name = an1}},
317   packagedElement = pi : uml::Package {name = 'Services'+pn,
318     packagedElement = se :uml::Package{name = cn+'Service',
319 packagedElement = sin:uml::Class{name=cn},
320     packagedElement = col:uml::Collaboration{name=cn,
321       ownedAttribute=ot:uml::Property{name='provider',type=sin,
322         aggregation=uml::AggregationKind::composite},
323       ownedAttribute=ot1:uml::Property{name='consumer',type=sin,
324         aggregation=uml::AggregationKind::composite}}}}};
325 enforce domain soaml mo:uml::Model{name = pn,
326 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
327 packagedElement = uma:uml::Collaboration {name = bn,
328   ownedAttribute = oa:uml::Property{name = an, type = pae,
329   aggregation = uml::AggregationKind::composite},
330   ownedAttribute = oa1:uml::Property{name = an1, type = pae1,
331   aggregation = uml::AggregationKind::composite},
332   collaborationUse = cu:uml::CollaborationUse {name = cn, type = col,
333   roleBinding = rb:uml::Dependency{supplier = oa:uml::Property{},
334   client = ot:uml::Property{},client = cu:uml::CollaborationUse{}},
335   roleBinding = rb1:uml::Dependency{supplier = oa1:uml::Property{type = pae1},
336   client = ot1:uml::Property{}, client = cu:uml::CollaborationUse{type = col}}}}}}};
337 enforce domain soaml su:SoaML2::ServicesArchitecture{base_Collaboration = uma};
338 when {
339   ElementsToSoaMLElements(st,sr,co,se,si,sint,pro,cons,con,conn);
340 }
341 }
342 }

```

### C.2.3. Unidirectional option with simple UML Interfaces

The unidirectional option with simple UML Interfaces generates one interface for the service provider with the corresponding operation, parameters and messages for each one. The Service Port is typed with the interface of the provider and the Request Port with the conjugated. The Model, Participants, Messages, Services, ServiceContract and ServicesArchitecture are generated accordingly.

```

1 transformation bpmn2uml2soaml2 (bpmn:bpmn2, soaml:SoaML2) {
2 key uml::Interface {name};
3 key uml::Collaboration {name};
4 key uml::Class {name};
5 key uml::Package {name};
6 key uml::Property {name,type};
7 key uml::Property {name,default};
8 key uml::Operation {name};
9 key uml::Connector{end};
10 key uml::ConnectorEnd {role};
11 key uml::CollaborationUse{roleBinding};
12 key uml::Dependency {supplier,client};
13 key uml::Parameter {name};
14 key uml::Port {name};
15 /*top relation to define a SoaML model from the BPMN2 model of the BP*/
16 top relation ModelToSoaMLModel {
17 pn:String;
18 checkonly domain bpmn de:bpmn2::Definitions {name = pn};
19 enforce domain soaml mo:uml::Model{name = pn,
20 profileApplication = proa:uml::ProfileApplication{
21   appliedProfile = aproa:uml::Profile{
22     name = 'http://www.soaml.org/SoaML/1.0.0'}}};
23 enforce domain soaml sop:SoaML2::SoaMLPackage{base_Package = mo};
24 }
25 /*top relation to define SoaML Participans from BPMN2 pools (process) of the BP*/
26 top relation ParticipantToSoaMLParticipant{
27 pn,pp:String;
28 sop:SoaML2::SoaMLPackage;
29 sa:SoaML2::Participant;
30 checkonly domain bpmn de:bpmn2::Definitions{name=pn,
31 rootElements=po:bpmn2::Process{name = pp}};
32 enforce domain soaml mo:uml::Model{name = pn,
33 packagedElement = pa:uml::Package{name = 'Participants'+pn,
34 packagedElement = uma:uml::Class {name = pp}}};
35 enforce domain soaml sap:SoaML2::SoaMLPackage{
36 base_Package = pa};
37 when {
38   ModelToSoaMLModel(de,mo,sop);
39 }
40 where {
41   ParticipantsSoaMLrefs(po,uma,sa) or true;
42 }
43 }
44 relation ParticipantsSoaMLrefs{
45   pp:String;
46 checkonly domain bpmn po:bpmn2::Process{name = pp};
47 enforce domain soaml uma:uml::Class {name = pp};
48 enforce domain soaml sa:SoaML2::Participant{base_Class = uma};
49 }
50 /*top relation to define SoaML MessagesTypes from BPMN2 MessageFlows of the BP*/
51 top relation MessageToSoaMLMessage{
52 pn,cn,sn:String;
53 sop:SoaML2::SoaMLPackage;
54 se:SoaML2::MessageType;
55 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
56 rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},

```

```

57 rootElements=p1:bpnm2::Process{flowElements=sr:bpnm2::Task{name = sn}},
58 rootElements=co:bpnm2::Collaboration{messageFlows=
59 me:bpnm2::MessageFlow{targetRef=st,sourceRef=sr}}};
60   enforce domain soaml mo:uml::Model{name = pn,
61   packagedElement = pe:uml::Package{name = 'Messages'+pn},
62   packagedElement = ume:uml::Class {name = cn+'Message'}}};
63   enforce domain soaml sep:SoaML2::SoaMLPackage{base_Package = pe};
64   when {
65   ModelToSoaMLModel(de,mo,sop);
66   }
67   where {
68   MessagesSoaMLrefs(po,p1,co,ume,se) or true;
69   }
70   }
71   relation MessagesSoaMLrefs{
72   sn,cn:String;
73   checkonly domain bpmn po:bpnm2::Process{flowElements=st:bpnm2::ServiceTask{name = cn}};
74   checkonly domain bpmn p1:bpnm2::Process{flowElements=sr:bpnm2::Task{name = sn}};
75   checkonly domain bpmn co:bpnm2::Collaboration
76   {messageFlows=me:bpnm2::MessageFlow{targetRef=st,sourceRef=sr}}};
77   checkonly domain soaml ume:uml::Class {name = cn+'Message'};
78   enforce domain soaml se:SoaML2::MessageType{base_Class = ume};
79   }
80   /*top relation to define SoaML Services from BPMN2 ServiceTasks, Tasks and MessageFlows of the
81   top relation ServicesToSoaMLServices{
82   pn,sn,cn:String;
83   pro:SoaML2::Provider;
84   con:SoaML2::ServiceContract;
85   conn:SoaML2::ServiceChannel;
86   si:SoaML2::SoaMLPackage;
87   sop:SoaML2::SoaMLPackage;
88   checkonly domain bpmn de:bpnm2::Definitions{name = pn,
89   rootElements=po:bpnm2::Process{flowElements=st:bpnm2::ServiceTask{name = cn}},
90   rootElements=p1:bpnm2::Process{flowElements=sr:bpnm2::Task{name = sn}},
91   rootElements=co:bpnm2::Collaboration{messageFlows=
92   me:bpnm2::MessageFlow{targetRef=st, sourceRef=sr}}};
93   enforce domain soaml mo:uml::Model{name = pn,
94   packagedElement = pi:uml::Package{name = 'Services'+pn,
95   packagedElement = se:uml::Package {name = cn+'Service'}}};
96   enforce domain soaml sip:SoaML2::SoaMLPackage{base_Package = pi};
97   when {
98   ModelToSoaMLModel(de,mo,sop);
99   }
100  where {
101      ElementsToSoaMLElements(st,sr,co,se,si,pro,con,conn) or true;
102  }
103  }
104  relation ElementsToSoaMLElements{
105  cn,sn:String;
106  checkonly domain bpmn st:bpnm2::ServiceTask{name = cn};
107  checkonly domain bpmn sr:bpnm2::Task{name = sn};
108  checkonly domain bpmn co:bpnm2::Collaboration{messageFlows=
109  me:bpnm2::MessageFlow{targetRef=st, sourceRef=sr}}};
110  enforce domain soaml se:uml::Package{name = cn+'Service',
111  packagedElement = it:uml::Interface{name= cn,
112  ownedOperation=owp:uml::Operation{name = cn+'OperationReceive'},

```

```

113     visibility=uml::VisibilityKind::public,
114     ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1In',
115     direction=uml::ParameterDirectionKind::inout},
116     ownedParameter = owpe:uml::Parameter{name=cn+'Return',
117     direction=uml::ParameterDirectionKind::return}}},
118 packagedElement = col:uml::Collaboration{ name=cn,
119     ownedAttribute=ot:uml::Property{name='provider',type=it,
120 aggregation=uml::AggregationKind::composite},
121     ownedAttribute=ot1:uml::Property{name='consumer',
122     default=sn,aggregation=uml::AggregationKind::composite},
123     ownedConnector = oc:uml::Connector{
124     end = en:uml::ConnectorEnd {role = ot,isUnique=false},
125 end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}}};
126 enforce domain soaml si:SoaML2::SoaMLPackage{base_Package = se};
127 enforce domain soaml pro:SoaML2::Provider{base_Interface = it};
128 enforce domain soaml con:SoaML2::ServiceContract{base_Collaboration = col};
129 enforce domain soaml conn:SoaML2::ServiceChannel {base_Connector = oc:uml::Connector{}};
130 }
131 /*top relation to update SoaML Services adding MessageTypes as the type of the parameters */
132 top relation MessageToServiceSoaMLMessage{
133     pn,cn,sn:String;
134     mes:SoaML2::MessageType;
135     si:SoaML2::SoaMLPackage;
136     pro:SoaML2::Provider;
137     con:SoaML2::ServiceContract;
138     conn:SoaML2::ServiceChannel;
139     checkonly domain bpmn de:bpmn2::Definitions{name = pn,
140     rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn}},
141     rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
142     rootElements=co:bpmn2::Collaboration{messageFlows=
143     me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
144     checkonly domain soaml mo:uml::Model{name = pn,
145     packagedElement = pe : uml::Package{name = 'Messages'+pn,
146     packagedElement = mae:uml::Class {name = cn+'Message'}},
147     packagedElement = pi : uml::Package {name = 'Services'+pn,
148     packagedElement = se :uml::Package{name = cn+'Service',
149     packagedElement = it:uml::Interface{name=cn,
150     ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
151     visibility=uml::VisibilityKind::public,
152     ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1In',
153     direction=uml::ParameterDirectionKind::inout},
154     ownedParameter = owpe:uml::Parameter{name=cn+'Return',
155     direction=uml::ParameterDirectionKind::return}}}}}}};
156     enforce domain soaml mo:uml::Model{name = pn,
157     packagedElement = pi : uml::Package {name = 'Services'+pn,
158     packagedElement = se :uml::Package{name = cn+'Service',
159     packagedElement = it:uml::Interface{name=cn,
160     ownedOperation=owp:uml::Operation{name = cn+'OperationReceive',
161     visibility=uml::VisibilityKind::public,
162     ownedParameter = owpa:uml::Parameter{name=cn+'Parameter1In',
163     direction=uml::ParameterDirectionKind::inout, type = mae},
164     ownedParameter = owpe:uml::Parameter{name=cn+'Return',
165     direction=uml::ParameterDirectionKind::return, type=mae}}}}}}};
166     when {
167     MessagesSoaMLrefs(po,p1,co,mae,mes);
168     ElementsToSoaMLElements(st,sr,co,se,si,pro,con,conn);

```

```

169   }
170 }
171 /*top relation to update SoaML Participants adding Ports corresponding*/
172 /*to generated services */
173 top relation PortToSoaMLPort{
174     pn,an,cn,an1,sn:String;
175 spart,spart1:SoaML2::Participant;
176 ser:SoaML2::Service;
177 req:SoaML2::Request;
178 pro:SoaML2::Provider;
179 con:SoaML2::ServiceContract;
180 conn:SoaML2::ServiceChannel;
181 si:SoaML2::SoaMLPackage;
182 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
183 rootElements=po:bpmn2::Process{name = an,
184 flowElements=st:bpmn2::ServiceTask{name = cn}},
185 rootElements=p1:bpmn2::Process{name = an1,
186 flowElements=sr:bpmn2::Task{name = sn}},
187 rootElements=co:bpmn2::Collaboration{messageFlows=
188 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
189 checkonly domain soaml mo:uml::Model{name = pn,
190 packagedElement = pa : uml::Package{name = 'Participants'+pn,
191 packagedElement = pae:uml::Class {name = an},
192 packagedElement = pae1:uml::Class {name = an1}},
193     packagedElement = pi : uml::Package {name = 'Services'+pn,
194         packagedElement = se :uml::Package{name = cn+'Service',
195             packagedElement = it:uml::Interface{name=cn}}}}};
196 enforce domain soaml mo:uml::Model{name = pn,
197 packagedElement = pa : uml::Package{name = 'Participants'+pn,
198 packagedElement = pae:uml::Class {name = an,
199     ownedAttribute = opor:uml::Port{ name = cn,
200         isService=true, type=it}},
201 packagedElement = pae1:uml::Class {name = an1,
202     ownedAttribute = opor1:uml::Port{name = sn,
203         isService=true, type=it,isConjugated=true}}}}};
204 when {
205     ParticipantsSoaMLrefs(po,pae,spart);
206     ParticipantsSoaMLrefs(p1,pae1,spart1);
207     ElementsToSoaMLElements(st,sr,co,se,si,pro,con,conn);
208 }
209 where {
210 PortSoaMLrefs(po,p1,co,pae,pae1,ser,req) or true;
211 }
212 }
213 relation PortSoaMLrefs {
214 an,cn,an1,sn:String;
215 checkonly domain bpmn po:bpmn2::Process{name = an,
216     flowElements=st:bpmn2::ServiceTask{name = cn}};
217 checkonly domain bpmn p1:bpmn2::Process{name = an1,
218     flowElements=sr:bpmn2::Task {name = sn}};
219     checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
220 me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}};
221 enforce domain soaml pae:uml::Class {name = an,
222     ownedAttribute = opor:uml::Port{name = cn} };
223 enforce domain soaml pae1:uml::Class {name = an1,
224     ownedAttribute = opor1:uml::Port{name = sn} };

```

```

225 enforce domain soaml ser:SoaML2::Service{base_Port = opor};
226 enforce domain soaml req:SoaML2::Request{base_Port = opor1};
227 }
228 /*top relation to define SoaML ServicesArchitecture from BPMN2 Collaboration*/
229 /*and generated SoaML elements */
230 top relation CollaborationToSoaMLServicesArchitecture{
231   pn,sn:String;
232   sop:SoaML2::SoaMLPackage;
233   su:SoaML2::ServicesArchitecture;
234   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
235     rootElements=co:bpmn2::Collaboration{name = sn} };
236   enforce domain soaml mo:uml::Model{name = pn,
237     packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
238     packagedElement = uma:uml::Collaboration {name = sn}}};
239     enforce domain soaml sup:SoaML2::SoaMLPackage{base_Package = pu };
240   when {
241     ModelToSoaMLModel(de,mo,sop);
242   }
243   where {
244     ServicesArchElementsToSoaMLServicesArchElements(de,mo,mo,su)or true;
245   }
246 }
247 relation ServicesArchElementsToSoaMLServicesArchElements{
248   pn,sn,bn,cn,an,an1:String;
249   pro:SoaML2::Provider;
250   con:SoaML2::ServiceContract;
251   conn:SoaML2::ServiceChannel;
252   si:SoaML2::SoaMLPackage;
253   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
254     rootElements=po:bpmn2::Process{name = an,
255     flowElements=st:bpmn2::ServiceTask{name = cn}},
256     rootElements=p1:bpmn2::Process{name = an1,
257     flowElements=sr:bpmn2::Task {name = sn}},
258     rootElements=co:bpmn2::Collaboration{name = bn,
259     participants = par:bpmn2::Participant{name = an, processRef = po},
260     participants = par1:bpmn2::Participant {name = an1, processRef = p1},
261     messageFlows = msf:bpmn2::MessageFlow{targetRef = st, sourceRef = sr}}};
262   checkonly domain soaml mo:uml::Model{name = pn,
263     packagedElement = pa : uml::Package{name = 'Participants'+pn,
264     packagedElement = pae:uml::Class {name = an},
265     packagedElement = pae1:uml::Class {name = an1}},
266     packagedElement = pi : uml::Package {name = 'Services'+pn,
267     packagedElement = se :uml::Package{name = cn+'Service',
268     packagedElement = it:uml::Interface{name=cn},
269     packagedElement = col:uml::Collaboration{name=cn,
270     ownedAttribute=ot:uml::Property{name='provider',type=it,
271     aggregation=uml::AggregationKind::composite},
272     ownedAttribute=ot1:uml::Property{name='consumer',
273     default=sn,aggregation=uml::AggregationKind::composite},
274     ownedConnector = oc:uml::Connector{
275     end = en:uml::ConnectorEnd {role = ot,isUnique=false},
276     end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}}}}};
277   enforce domain soaml mo:uml::Model{name = pn,
278     packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
279     packagedElement = uma:uml::Collaboration {name = bn,
280     ownedAttribute = oa:uml::Property{name = an, type = pae,

```

```

281     aggregation = uml::AggregationKind::composite},
282     ownedAttribute = oa1:uml::Property{name = an1, type = pae1,
283     aggregation = uml::AggregationKind::composite},
284     collaborationUse = cu:uml::CollaborationUse {name = cn, type = col,
285     roleBinding = rb:uml::Dependency{supplier = oa:uml::Property},
286     client = ot:uml::Property},
287     client = cu:uml::CollaborationUse}},
288     roleBinding = rb1:uml::Dependency{supplier = oa1:uml::Property{type = pae1},
289     client = ot1:uml::Property},
290     client = cu:uml::CollaborationUse{type = col}}}}}}};
291 enforce domain soaml su:SoaML2::ServicesArchitecture{base_Collaboration = uma};
292 when {
293     ElementsToSoaMLElements(st,sr,co,se,si,pro,con,conn);
294 }
295 }
296 }

```

### C.3. Generation from ServiceTask and other elements

In this section the code for the QVT transformations defined for the generation from ServiceTask plus Interface, Operation and Messages for the service provided identified in the BPMN2 model are presented: bidirectional option with simple UML Interfaces, bidirectional option with ServiceInterfaces and unidirectional option with simple UML Interfaces.

#### C.3.1. Bidirectional option with simple UML Interfaces

The bidirectional option with simple UML Interfaces generates the same elements as presented in subsection C.2.1 but as the Interface, Operation and Messages for the provider are also present in the BPMN2 model, these names are preserved in the generation of the corresponding ones in the SoaML model. The structure of the code is the same but the existence of these elements in the BPMN2 model are also checked and used as a basis for the names of the generation for the service provider. For the service consumer the generation is the same as in subsection C.2.1.

```

1 transformation bpmn2uml2soaml2 (bpmn:bpmn2, soaml:SoaML2) {
2 key uml::Interface {name};
3 key uml::Collaboration {name};
4 key uml::Class {name};
5 key uml::Package {name};
6 key uml::Property {name,type};
7 key uml::Operation {name};
8 key uml::Connector{end};
9 key uml::ConnectorEnd {role};
10 key uml::CollaborationUse{roleBinding};
11 key uml::Dependency {supplier,client};
12 key uml::Parameter {name};
13 key uml::Port {name};
14 /*top relation to define a SoaML model from the BPMN2 model of the BP*/
15 top relation ModelToSoaMLModel {
16     pn:String;
17 checkonly domain bpmn de:bpmn2::Definitions {name = pn};
18 enforce domain soaml mo:uml::Model{name = pn,
19     profileApplication = proa:uml::ProfileApplication{
20     appliedProfile = aproa:uml::Profile{
21     name = 'http://www.soaml.org/SoaML/1.0.0'}}}}};

```

```

22 enforce domain soaml sop:SoaML2::SoaMLPackage{base_Package = mo};
23 }
24 /*top relation to define SoaML Participans from BPMN2 pools (process) of the BP*/
25 top relation ParticipantToSoaMLParticipant{
26   pn,pp:String;
27   sop:SoaML2::SoaMLPackage;
28   sa:SoaML2::Participant;
29   checkonly domain bpmn de:bpmn2::Definitions{name=pn,
30     rootElements=po:bpmn2::Process{name = pp}};
31   enforce domain soaml mo:uml::Model{name = pn,
32     packagedElement = pa:uml::Package{name = 'Participants'+pn,
33     packagedElement = uma:uml::Class {name = pp}}};
34   enforce domain soaml sap:SoaML2::SoaMLPackage{
35     base_Package = pa};
36   when {
37     ModelToSoaMLModel(de,mo,sop);
38   }
39   where {
40     ParticipantsSoaMLrefs(po,uma,sa) or true;
41   }
42 }
43 relation ParticipantsSoaMLrefs{
44   pp:String;
45   checkonly domain bpmn po:bpmn2::Process{name = pp};
46   enforce domain soaml uma:uml::Class {name = pp};
47   enforce domain soaml sa:SoaML2::Participant{base_Class = uma};
48 }
49 /*top relation to define SoaML MessagesTypes from BPMN2 MessageFlows of the BP*/
50 top relation MessageToSoaMLMessage{
51   pn,cn,sn,bn,cn1,bn1:String;
52   sop:SoaML2::SoaMLPackage;
53   se,se1:SoaML2::MessageType;
54   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
55     rootElements=mec:bpmn2::Message{name=bn},
56     rootElements=incr:bpmn2::Interface{name=cn1,
57     operations=op:bpmn2::Operation{name = bn1,
58     inMessageRef=mec,outMessageRef=mec}},
59     rootElements=po:bpmn2::Process{flowElements=
60     st:bpmn2::ServiceTask{name=cn,operationRef=op}},
61     rootElements=p1:bpmn2::Process{
62     flowElements=sr:bpmn2::Task{name = sn}},
63     rootElements=co:bpmn2::Collaboration{messageFlows=
64     me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
65   enforce domain soaml mo:uml::Model{name = pn,
66     packagedElement = pe:uml::Package{name = 'Messages'+pn,
67     packagedElement = ume:uml::Class {name = bn+'Message'},
68     packagedElement = ume1:uml::Class {name = sn+'Message'}}};
69   enforce domain soaml sep:SoaML2::SoaMLPackage{base_Package = pe};
70   when {
71     ModelToSoaMLModel(de,mo,sop);
72   }
73   where {
74     MessagesSoaMLrefs(mec,incr,po,p1,co,ume,ume1,se,se1) or true;
75   }
76 }
77 relation MessagesSoaMLrefs{

```



```

78 sn,bn,bn1,cn1,cn:String;
79 checkonly domain bpmn mec:bpmn2::Message{name = bn};
80 checkonly domain bpmn inr:bpmn2::Interface{name=cn1,
81 operations=op:bpmn2::Operation{name = bn1,
82   inMessageRef=mec,outMessageRef=mec}}};
83   checkonly domain bpmn po:bpmn2::Process{
84     flowElements=st:bpmn2::ServiceTask{name = cn, operationRef=op}}};
85   checkonly domain bpmn p1:bpmn2::Process{
86     flowElements=sr:bpmn2::Task{name = sn}}};
87 checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
88 me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
89 checkonly domain soaml ume:uml::Class {name = bn+'Message'};
90 checkonly domain soaml ume1:uml::Class {name = sn+'Message'};
91 enforce domain soaml se:SoaML2::MessageType{base_Class = ume};
92 enforce domain soaml se1:SoaML2::MessageType{base_Class = ume1};
93   }
94 /*top relation to define SoaML Services from BPMN2 ServiceTasks, Tasks */
95 /*and MessageFlows of the BP*/
96 top relation ServicesToSoaMLServices{
97 pn,sn,cn,bn,cn1,bn1:String;
98 pro:SoaML2::Provider;
99 cons:SoaML2::Consumer;
100 con:SoaML2::ServiceContract;
101 conn:SoaML2::ServiceChannel;
102 si:SoaML2::SoaMLPackage;
103 sop:SoaML2::SoaMLPackage;
104 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
105   rootElements=mec:bpmn2::Message{name=bn},
106   rootElements=inr:bpmn2::Interface{name=cn1,
107   operations=op:bpmn2::Operation{name = bn1,
108   inMessageRef=mec,outMessageRef=mec}}},
109 rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn,
110 operationRef=op}}},
111 rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
112 rootElements=co:bpmn2::Collaboration{messageFlows=
113 me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
114 enforce domain soaml mo:uml::Model{name = pn,
115 packagedElement = pi:uml::Package{name = 'Services'+pn,
116   packagedElement = se:uml::Package {name = cn1+'Service'}}};
117 enforce domain soaml sip:SoaML2::SoaMLPackage{base_Package = pi};
118 when {
119 ModelToSoaMLModel(de,mo,sop);
120 }
121 where {
122   ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,pro,cons,con,conn) or true;
123 }
124 }
125 relation ElementsToSoaMLElements{
126 cn,sn,bn,cn1,bn1:String;
127 checkonly domain bpmn mec:bpmn2::Message{name=bn};
128 checkonly domain bpmn inr:bpmn2::Interface{name=cn1,
129 operations=op:bpmn2::Operation{name = bn1,
130   inMessageRef=mec,outMessageRef=mec}}};
131 checkonly domain bpmn st:bpmn2::ServiceTask{name = cn, operationRef=op}}};
132 checkonly domain bpmn sr:bpmn2::Task{name = sn};
133 checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=

```

```

134 me:bpnm2::MessageFlow{targetRef=st, sourceRef=sr}};
135 enforce domain soaml se:uml::Package{name = cn1+'Service',
136 packagedElement = it:uml::Interface{name= cn1,
137   ownedOperation=owp:uml::Operation{name = bn1,
138   visibility=uml::VisibilityKind::public,
139   ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1',
140   direction=uml::ParameterDirectionKind::inout},
141   ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
142   direction=uml::ParameterDirectionKind::return}}},
143   packagedElement = it1:uml::Interface{name = sn,
144   ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
145   visibility=uml::VisibilityKind::public,
146   ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
147   direction=uml::ParameterDirectionKind::inout},
148   ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
149   direction=uml::ParameterDirectionKind::return}}},
150 packagedElement = us:uml::Usage{supplier = it:uml::Interface{name=cn1},
151 client = it1:uml::Interface{name=sn}},
152 packagedElement = us1:uml::Usage{supplier = it1:uml::Interface{name=sn},
153 client = it:uml::Interface{name=cn1}},
154 packagedElement = col:uml::Collaboration{ name=cn1+'ServiceContract',
155   ownedAttribute=ot:uml::Property{name='provider',type=it,
156   aggregation=uml::AggregationKind::composite},
157   ownedAttribute=ot1:uml::Property{name='consumer', type=it1,
158   aggregation=uml::AggregationKind::composite},
159 ownedConnector = oc:uml::Connector{
160 end = en:uml::ConnectorEnd {role = ot,isUnique=false},
161   end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}};
162   enforce domain soaml si:SoaML2::SoaMLPackage{base_Package = se};
163   enforce domain soaml pro:SoaML2::Provider{base_Interface = it};
164   enforce domain soaml cons:SoaML2::Consumer{base_Interface = it1};
165   enforce domain soaml con:SoaML2::ServiceContract{base_Collaboration = col};
166   enforce domain soaml conn:SoaML2::ServiceChannel {base_Connector = oc:uml::Connector{}};
167   }
168 /*top relation to update SoaML Services adding MessageTypes as the*/
169 /*type of the parameters */
170   top relation MessageToServiceSoaMLMessage{
171   pn,bn,an,cn,an1,sn,cn1,bn1:String;
172   mes,mes1:SoaML2::MessageType;
173 si:SoaML2::SoaMLPackage;
174 pro:SoaML2::Provider;
175 cons:SoaML2::Consumer;
176 con:SoaML2::ServiceContract;
177 conn:SoaML2::ServiceChannel;
178   checkonly domain bpmn de:bpnm2::Definitions{name = pn,
179 rootElements=mec:bpnm2::Message{name=bn},
180   rootElements=inr:bpnm2::Interface{name=cn1,
181   operations=op:bpnm2::Operation{name = bn1,
182   inMessageRef=mec,outMessageRef=mec}},
183 rootElements=po:bpnm2::Process{flowElements=st:bpnm2::ServiceTask{name = cn,
184 operationRef=op}},
185 rootElements=p1:bpnm2::Process{flowElements=sr:bpnm2::Task{name = sn}},
186 rootElements=co:bpnm2::Collaboration{messageFlows=
187 me:bpnm2::MessageFlow{targetRef=st, sourceRef=sr}}};
188 checkonly domain soaml mo:uml::Model{name = pn,
189 packagedElement = pe : uml::Package{name = 'Messages'+pn,

```

```

190 packagedElement = mae:uml::Class {name = bn+'Message'},
191 packagedElement = mae1:uml::Class {name = sn+'Message'}},
192 packagedElement = pi : uml::Package {name = 'Services'+pn,
193 packagedElement = se :uml::Package{name = cn1+'Service',
194 packagedElement = it:uml::Interface{name=cn1,
195   ownedOperation=owp:uml::Operation{name = bn1,
196   visibility=uml::VisibilityKind::public,
197 ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1',
198 direction=uml::ParameterDirectionKind::inout},
199   ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
200   direction=uml::ParameterDirectionKind::return}}},
201 packagedElement = it1:uml::Interface{name=sn,
202 ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
203 visibility=uml::VisibilityKind::public,
204 ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
205 direction=uml::ParameterDirectionKind::inout},
206   ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
207   direction=uml::ParameterDirectionKind::return}}}}}}};
208 enforce domain soaml mo:uml::Model{name = pn,
209   packagedElement = pi : uml::Package {name = 'Services'+pn,
210   packagedElement = se :uml::Package{name = cn1+'Service',
211   packagedElement = it:uml::Interface{name=cn1,
212   ownedOperation=owp:uml::Operation{name = bn1,
213   visibility=uml::VisibilityKind::public,
214   ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1',
215   direction=uml::ParameterDirectionKind::inout, type = mae},
216   ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
217   direction=uml::ParameterDirectionKind::return, type = mae}}},
218   packagedElement = it1:uml::Interface{name=sn,
219   ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
220   visibility=uml::VisibilityKind::public,
221   ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
222   direction=uml::ParameterDirectionKind::inout, type = mae1},
223   ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
224   direction=uml::ParameterDirectionKind::return, type = mae1}}}}}}};
225 when {
226   MessagesSoAMLrefs(mec,inr,po,p1,co,mae,mae1,mes,mes1);
227   ElementsToSoAMLElements(mec,inr,st,sr,co,se,si,pro,cons,con,conn);
228 }
229 }
230 /*top relation to update SoaML Participants adding Ports corresponding*/
231 /*to generated services */
232 top relation PortToSoaMLPort{
233   pn,bn,an,cn,an1,sn,cn1,bn1,bn2:String;
234   spart,spart1:SoaML2::Participant;
235   ser:SoaML2::Service;
236   req:SoaML2::Request;
237   pro:SoaML2::Provider;
238   cons:SoaML2::Consumer;
239   con:SoaML2::ServiceContract;
240   conn:SoaML2::ServiceChannel;
241   si:SoaML2::SoaMLPackage;
242   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
243   rootElements=mec:bpmn2::Message{name=bn},
244   rootElements=inr:bpmn2::Interface{name=cn1,
245   operations=op:bpmn2::Operation{name = bn1,

```

```

246   inMessageRef=mec,outMessageRef=mec}},
247   rootElements=po:bpnm2::Process{name = an,
248   flowElements=st:bpnm2::ServiceTask{name = cn,operationRef=op}},
249   rootElements=p1:bpnm2::Process{name = an1,
250   flowElements=sr:bpnm2::Task {name = sn}},
251   rootElements=co:bpnm2::Collaboration{
252   messageFlows = msf:bpnm2::MessageFlow{targetRef = st, sourceRef = sr}}};
253 checkonly domain soaml mo:uml::Model{name = pn,
254 packagedElement = pa : uml::Package{name = 'Participants'+pn,
255 packagedElement = pae:uml::Class {name = an},
256 packagedElement = pae1:uml::Class {name = an1}},
257 packagedElement = pi : uml::Package {name = 'Services'+pn,
258   packagedElement = se :uml::Package{name = cn1+'Service',
259   packagedElement = it:uml::Interface{name=cn1},
260   packagedElement = it1:uml::Interface{name=sn}}}}};
261 enforce domain soaml mo:uml::Model{name = pn,
262 packagedElement = pa : uml::Package{name = 'Participants'+pn,
263 packagedElement = pae:uml::Class {name = an,
264   ownedAttribute = opor:uml::Port{name = cn1,
265   isService=true, type=it}},
266 packagedElement = pae1:uml::Class {name = an1,
267   ownedAttribute = opor1:uml::Port{name = sn,
268   isService=true, type=it1}} } };
269 when {
270   ParticipantsSoaMLrefs(po,pae,spart);
271   ParticipantsSoaMLrefs(p1,pae1,spart1);
272   ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,pro,cons,con,conn);
273 }
274 where {
275 PortSoaMLrefs(mec,inr,po,p1,co,pae,pae1,ser,req) or true;
276 }
277 }
278 relation PortSoaMLrefs {
279 an,cn,an1,sn,cn1,bn,bn1:String;
280 checkonly domain bpmn mec:bpnm2::Message{name=bn};
281 checkonly domain bpmn inr:bpnm2::Interface{name=cn1,
282 operations=op:bpnm2::Operation{name = bn1,
283   inMessageRef=mec,outMessageRef=mec}};
284 checkonly domain bpmn po:bpnm2::Process{name = an,
285   flowElements=st:bpnm2::ServiceTask{name = cn, operationRef=op}};
286 checkonly domain bpmn p1:bpnm2::Process{name = an1,
287 flowElements=sr:bpnm2::Task {name = sn}};
288 checkonly domain bpmn co:bpnm2::Collaboration{messageFlows=
289 me:bpnm2::MessageFlow{targetRef=st, sourceRef=sr}};
290 enforce domain soaml pae:uml::Class {name = an,
291   ownedAttribute = opor:uml::Port{name = cn1}};
292 enforce domain soaml pae1:uml::Class {name = an1,
293   ownedAttribute = opor1:uml::Port{name = sn}};
294 enforce domain soaml ser:SoaML2::Service{base_Port = opor};
295 enforce domain soaml req:SoaML2::Request{base_Port = opor1};
296 }
297 /*top relation to define SoaML ServicesArchitecture from BPMN2 Collaboration */
298 /*and generated SoaML elements */
299 top relation CollaborationToSoaMLServicesArchitecture{
300 pn,bn2:String;
301 sop:SoaML2::SoaMLPackage;

```

```

302 su:SoaML2::ServicesArchitecture;
303 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
304 rootElements=co:bpmn2::Collaboration{name = bn2}};
305 enforce domain soaml mo:uml::Model{name = pn,
306 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
307 packagedElement = uma:uml::Collaboration {name = bn2}}};
308   enforce domain soaml sup:SoaML2::SoaMLPackage{base_Package = pu};
309 when {
310 ModelToSoaMLModel(de,mo,sop);
311 }
312 where {
313 ServicesArchElementsToSoaMLServicesArchElements(de,mo,mo,su) or true;
314 }
315 }
316 relation ServicesArchElementsToSoaMLServicesArchElements{
317 pn,sn,bn,cn,an,an1,cn1,bn1,bn2:String;
318 pro:SoaML2::Provider;
319 cons:SoaML2::Consumer;
320 con:SoaML2::ServiceContract;
321 conn:SoaML2::ServiceChannel;
322 si:SoaML2::SoaMLPackage;
323 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
324   rootElements=mec:bpmn2::Message{name=bn},
325   rootElements=incr:bpmn2::Interface{name=cn1,
326   operations=op:bpmn2::Operation{name = bn1,
327   inMessageRef=mec,outMessageRef=mec}},
328   rootElements=po:bpmn2::Process{name = an,
329   flowElements=st:bpmn2::ServiceTask{name = cn,operationRef=op}},
330   rootElements=p1:bpmn2::Process{name = an1,
331   flowElements=sr:bpmn2::Task {name = sn}},
332   rootElements=co:bpmn2::Collaboration{name = bn2,
333   messageFlows = msf:bpmn2::MessageFlow{targetRef = st,sourceRef = sr},
334   participants = par:bpmn2::Participant{name = an, processRef = po},
335   participants = par1:bpmn2::Participant {name = an1, processRef = p1}}};
336 checkonly domain soaml mo:uml::Model{name = pn,
337 packagedElement = pa : uml::Package{name = 'Participants'+pn,
338 packagedElement = pae:uml::Class {name = an},
339 packagedElement = pae1:uml::Class {name = an1}},
340 packagedElement = pi : uml::Package {name = 'Services'+pn,
341   packagedElement = se :uml::Package{name = cn1+'Service',
342 packagedElement = it:uml::Interface{name=cn1},
343 packagedElement = it1:uml::Interface{name=sn},
344   packagedElement = col:uml::Collaboration{name=cn1+'ServiceContract',
345 ownedAttribute=ot:uml::Property{name='provider',type=it,
346   aggregation=uml::AggregationKind::composite},
347   ownedAttribute=ot1:uml::Property{name='consumer',type=it1,
348   aggregation=uml::AggregationKind::composite}}}}};
349 enforce domain soaml mo:uml::Model{name = pn,
350 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
351 packagedElement = uma:uml::Collaboration {name = bn2,
352   ownedAttribute = oa:uml::Property{name = an, type = pae,
353   aggregation = uml::AggregationKind::composite},
354   ownedAttribute = oa1:uml::Property{name = an1, type = pae1,
355   aggregation = uml::AggregationKind::composite},
356   collaborationUse = cu:uml::CollaborationUse {name = cn1, type = col,
357   roleBinding = rb:uml::Dependency{supplier = oa:uml::Property{}}},

```

```

358     client = ot:uml::Property{}, client = cu:uml::CollaborationUse{}},
359     roleBinding = rb1:uml::Dependency{supplier = oa1:uml::Property{type = pae1},
360     client = ot1:uml::Property{}, client = cu:uml::CollaborationUse{type = col}}}}}}};
361 enforce domain soaml su:SoaML2::ServicesArchitecture{base_Collaboration = uma};
362 when {
363     ElementsToSoaMLElements(mec, inr, st, sr, co, se, si, pro, cons, con, conn);
364 }
365 }
366 }

```

### C.3.2. Bidirectional option with ServiceInterfaces

The bidirectional option with ServiceInterfaces generates the same elements as presented in subsection C.2.2 adding the generation of the corresponding ServiceInterface and preserving the names for the Interface, Operation and Messages identified for the service provider, as presented in subsection C.3.1.

```

1 transformation bpmn2uml2soaml2 (bpmn:bpmn2, soaml:SoaML2) {
2 key uml::Interface {name};
3 key uml::Collaboration {name};
4 key uml::Class {name};
5 key uml::Package {name};
6 key uml::Property {name,type};
7 key uml::Operation {name};
8 key uml::Connector{end};
9 key uml::ConnectorEnd {role};
10 key uml::CollaborationUse{roleBinding};
11 key uml::Dependency {supplier,client};
12 key uml::Parameter {name};
13 key uml::Port {name};
14 /*top relation to define a SoaML model from the BPMN2 model of the BP*/
15 top relation ModelToSoaMLModel {
16     pn:String;
17 checkonly domain bpmn de:bpmn2::Definitions {name = pn};
18 enforce domain soaml mo:uml::Model{name = pn,
19     profileApplication = proa:uml::ProfileApplication{
20     appliedProfile = aproa:uml::Profile{
21     name = 'http://www.soaml.org/SoaML/1.0.0'}}}}};
22 enforce domain soaml sop:SoaML2::SoaMLPackage{base_Package = mo};
23 }
24 /*top relation to define SoaML Participans from BPMN2 pools (process) of the BP*/
25 top relation ParticipantToSoaMLParticipant{
26 pn,pp:String;
27 sop:SoaML2::SoaMLPackage;
28 sa:SoaML2::Participant;
29 checkonly domain bpmn de:bpmn2::Definitions{name=pn,
30 rootElements=po:bpmn2::Process{name = pp}}};
31 enforce domain soaml mo:uml::Model{name = pn,
32 packagedElement = pa:uml::Package{name = 'Participants'+pn,
33 packagedElement = uma:uml::Class {name = pp}}}}};
34 enforce domain soaml sap:SoaML2::SoaMLPackage{base_Package = pa};
35 when {
36 ModelToSoaMLModel(de,mo,sop);
37 }
38 where {
39     ParticipantsSoaMLrefs(po,uma,sa) or true;

```

```

40  }
41  }
42  relation ParticipantsSoaMLrefs{
43  pp:String;
44  checkonly domain bpmn po:bpmn2::Process{name = pp};
45  enforce domain soaml uma:uml::Class {name = pp};
46  enforce domain soaml sa:SoaML2::Participant{base_Class = uma};
47  }
48  /*top relation to define SoaML MessagesTypes from BPMN2 MessageFlows of the BP*/
49  top relation MessageToSoaMLMessage{
50  pn,cn,sn,bn,cn1,bn1:String;
51  sop:SoaML2::SoaMLPackage;
52  se,se1:SoaML2::MessageType;
53  checkonly domain bpmn de:bpmn2::Definitions{name = pn,
54  rootElements=mec:bpmn2::Message{name=bn},
55  rootElements=incr:bpmn2::Interface{name=cn1,
56  operations=op:bpmn2::Operation{name = bn1,
57  inMessageRef=mec,outMessageRef=mec}},
58  rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn,
59  operationRef=op}},
60  rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
61  rootElements=co:bpmn2::Collaboration{messageFlows=
62  me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
63  enforce domain soaml mo:uml::Model{name = pn,
64  packagedElement = pe:uml::Package{name = 'Messages'+pn,
65  packagedElement = ume:uml::Class {name = bn+'Message'},
66  packagedElement = ume1:uml::Class {name = sn+'Message'}}};
67  enforce domain soaml sep:SoaML2::SoaMLPackage{base_Package = pe};
68  when {
69  ModelToSoaMLModel(de,mo,sop);
70  }
71  where {
72  MessagesSoaMLrefs(mec,incr,po,p1,co,ume,ume1,se,se1) or true;
73  }
74  }
75  relation MessagesSoaMLrefs{
76  sn,bn,bn1,cn1,cn:String;
77  checkonly domain bpmn mec:bpmn2::Message{name = bn};
78  checkonly domain bpmn incr:bpmn2::Interface{name=cn1,
79  operations=op:bpmn2::Operation{name = bn1,
80  inMessageRef=mec,outMessageRef=mec}};
81  checkonly domain bpmn po:bpmn2::Process{
82  flowElements=st:bpmn2::ServiceTask{name = cn, operationRef=op}};
83  checkonly domain bpmn p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}};
84  checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
85  me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}};
86  checkonly domain soaml ume:uml::Class {name = bn+'Message'};
87  checkonly domain soaml ume1:uml::Class {name = sn+'Message'};
88  enforce domain soaml se:SoaML2::MessageType{base_Class = ume};
89  enforce domain soaml se1:SoaML2::MessageType{base_Class = ume1};
90  }
91  /*top relation to define SoaML Services from BPMN2 ServiceTasks, Tasks and MessageFlows of the
92  top relation ServicesToSoaMLServices{
93  pn,sn,cn,bn,cn1,bn1:String;
94  sint:SoaML2::ServiceInterface;
95  pro:SoaML2::Provider;

```

```

96 cons:SoaML2::Consumer;
97 con:SoaML2::ServiceContract;
98 conn:SoaML2::ServiceChannel;
99 si:SoaML2::SoaMLPackage;
100 sop:SoaML2::SoaMLPackage;
101 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
102   rootElements=mec:bpmn2::Message{name=bn},
103   rootElements=incr:bpmn2::Interface{name=cn1,
104     operations=op:bpmn2::Operation{name = bn1,
105       inMessageRef=mec,outMessageRef=mec}},
106   rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn,
107     operationRef=op}},
108   rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
109   rootElements=co:bpmn2::Collaboration{messageFlows=
110     me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
111 enforce domain soaml mo:uml::Model{name = pn,
112   packagedElement = pi:uml::Package{name = 'Services'+pn,
113     packagedElement = se:uml::Package {name = cn1+'Service'}}};
114 enforce domain soaml sip:SoaML2::SoaMLPackage{base_Package = pi};
115 when {
116   ModelToSoaMLModel(de,mo,sop);
117 }
118 where {
119   ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,sint,pro,cons,con,conn) or true;
120 }
121 }
122 relation ElementsToSoaMLElements{
123   cn,sn,bn,bn1,cn1:String;
124   checkonly domain bpmn mec:bpmn2::Message{name=bn};
125   checkonly domain bpmn incr:bpmn2::Interface{name=cn1,
126     operations=op:bpmn2::Operation{name = bn1,
127       inMessageRef=mec,outMessageRef=mec}};
128   checkonly domain bpmn st:bpmn2::ServiceTask{name = cn, operationRef=op};
129   checkonly domain bpmn sr:bpmn2::Task{name = sn};
130   checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
131     me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}};
132   enforce domain soaml se:uml::Package{name = cn1+'Service',
133     packagedElement = sin:uml::Class{name=cn1,
134       clientDependency = cd:uml::Dependency{
135         supplier = us:uml::Usage{}, supplier=ir:uml::InterfaceRealization{}},
136         interfaceRealization=ir:uml::InterfaceRealization{
137           supplier=it:uml::Interface{}, contract=it:uml::Interface{}, client=sin:uml::Class{}},
138         packagedElement = us:uml::Usage{supplier = it1:uml::Interface{}, client = sin:uml::Class{}},
139         packagedElement = it:uml::Interface{name= cn1,
140           ownedOperation=owp:uml::Operation{name = bn1,
141             visibility=uml::VisibilityKind::public,
142             ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1',
143               direction=uml::ParameterDirectionKind::inout},
144             ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
145               direction=uml::ParameterDirectionKind::return}}},
146           packagedElement = it1:uml::Interface{name = sn,
147             ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
148               visibility=uml::VisibilityKind::public,
149               ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
150                 direction=uml::ParameterDirectionKind::inout},
151               ownedParameter = owpe1:uml::Parameter{name=sn+'Return',

```



```

152     direction=uml::ParameterDirectionKind::return}}},
153 packagedElement = col:uml::Collaboration{ name=cn1,
154     ownedAttribute=ot:uml::Property{name='provider',type=sin,
155     aggregation=uml::AggregationKind::composite},
156     ownedAttribute=ot1:uml::Property{name='consumer', type=sin,
157     aggregation=uml::AggregationKind::composite},
158     ownedConnector = oc:uml::Connector{
159     end = en:uml::ConnectorEnd {role = ot,isUnique=false},
160     end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}};
161     enforce domain soaml si:SoaML2::SoaMLPackage{base_Package = se};
162     enforce domain soaml sint:SoaML2::ServiceInterface{base_Class = sin};
163     enforce domain soaml pro:SoaML2::Provider{base_Interface = it};
164     enforce domain soaml cons:SoaML2::Consumer{base_Interface = it1};
165     enforce domain soaml con:SoaML2::ServiceContract{base_Collaboration = col};
166     enforce domain soaml conn:SoaML2::ServiceChannel {base_Connector = oc:uml::Connector{}};
167 }
168 /*top relation to update SoaML Services adding MessageTypes as*/
169 /*the type of the parameters */
170     top relation MessageToServiceSoaMLMessage{
171     pn,bn,an,cn,an1,sn,cn1,bn1:String;
172     tn,dn,tn1,en:String;
173     mes,mes1:SoaML2::MessageType;
174     si:SoaML2::SoaMLPackage;
175     sint:SoaML2::ServiceInterface;
176     pro:SoaML2::Provider;
177     cons:SoaML2::Consumer;
178     con:SoaML2::ServiceContract;
179     conn:SoaML2::ServiceChannel;
180     checkonly domain bpmn de:bpmn2::Definitions{name = pn,
181     rootElements=mec:bpmn2::Message{name=bn},
182     rootElements=incr:bpmn2::Interface{name=cn1,
183     operations=op:bpmn2::Operation{name = bn1,
184     inMessageRef=mec,outMessageRef=mec}},
185     rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn,
186     operationRef=op}},
187     rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
188     rootElements=co:bpmn2::Collaboration{messageFlows=
189     me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
190     checkonly domain soaml mo:uml::Model{name = pn,
191     packagedElement = pe : uml::Package{name = 'Messages'+pn,
192     packagedElement = mae:uml::Class {name = bn+'Message'},
193     packagedElement = mae1:uml::Class {name = sn+'Message'}},
194     packagedElement = pi : uml::Package {name = 'Services'+pn,
195     packagedElement = se :uml::Package{name = cn1+'Service',
196     packagedElement = it:uml::Interface{name=cn1,
197     ownedOperation=owp:uml::Operation{name = bn1,
198     visibility=uml::VisibilityKind::public,
199     ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1',
200     direction=uml::ParameterDirectionKind::inout},
201     ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
202     direction=uml::ParameterDirectionKind::return}}},
203     packagedElement = it1:uml::Interface{name=sn,
204     ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
205     visibility=uml::VisibilityKind::public,
206     ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
207     direction=uml::ParameterDirectionKind::inout},

```

```

208 ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
209 direction=uml::ParameterDirectionKind::return}}}}}};
210 enforce domain soaml mo:uml::Model{name = pn,
211   packagedElement = pi : uml::Package {name = 'Services'+pn,
212     packagedElement = se :uml::Package{name = cn1+'Service',
213   packagedElement = it:uml::Interface{name=cn1,
214     ownedOperation=owp:uml::Operation{name = bn1,
215     visibility=uml::VisibilityKind::public,
216   ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1',
217   direction=uml::ParameterDirectionKind::inout, type = mae},
218   ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
219   direction=uml::ParameterDirectionKind::return, type = mae}}},
220   packagedElement = it1:uml::Interface{name=sn,
221   ownedOperation=owp1:uml::Operation{name = sn+'OperationSend',
222   visibility=uml::VisibilityKind::public,
223   ownedParameter = owpa1:uml::Parameter{name=sn+'Parameter1',
224   direction=uml::ParameterDirectionKind::inout, type = mae1},
225   ownedParameter = owpe1:uml::Parameter{name=sn+'Return',
226   direction=uml::ParameterDirectionKind::return, type = mae1}}}}}}};
227 when {
228   MessagesSoaMLrefs(mec,inr,po,p1,co,mae,mae1,mes,mes1);
229   ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,sint,pro,cons,con,conn);
230 }
231 }
232 /*top relation to update SoaML Participants adding Ports corresponding*/
233 /*to generated services */
234 top relation PortToSoaMLPort{
235   pn,bn,an,cn,an1,sn,bn1,cn1,bn2:String;
236   spart,spart1:SoaML2::Participant;
237   ser:SoaML2::Service;
238   req:SoaML2::Request;
239   sint:SoaML2::ServiceInterface;
240   pro:SoaML2::Provider;
241   cons:SoaML2::Consumer;
242   con:SoaML2::ServiceContract;
243   conn:SoaML2::ServiceChannel;
244   si:SoaML2::SoaMLPackage;
245   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
246     rootElements=mec:bpmn2::Message{name=bn},
247     rootElements=inr:bpmn2::Interface{name=cn1,
248     operations=op:bpmn2::Operation{name = bn1,
249     inMessageRef=mec,outMessageRef=mec}},
250     rootElements=po:bpmn2::Process{name = an, flowElements=st:bpmn2::ServiceTask{name=cn,
251     operationRef=op}},
252     rootElements=p1:bpmn2::Process{name = an1, flowElements=sr:bpmn2::Task{name=sn}},
253     rootElements=co:bpmn2::Collaboration{
254     messageFlows = msf:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
255   checkonly domain soaml mo:uml::Model{name = pn,
256   packagedElement = pa : uml::Package{name = 'Participants'+pn,
257   packagedElement = pae:uml::Class {name = an},
258   packagedElement = pae1:uml::Class {name = an1}},
259   packagedElement = pi : uml::Package {name = 'Services'+pn,
260     packagedElement = se :uml::Package{name = cn1+'Service',
261   packagedElement = sin:uml::Class{name=cn1}}}}};
262   enforce domain soaml mo:uml::Model{name = pn,
263   packagedElement = pa : uml::Package{name = 'Participants'+pn,

```

```

264 packagedElement = pae:uml::Class {name = an,
265   ownedAttribute = opor:uml::Port{name = cn1,
266     isService=true, type=sin}},
267 packagedElement = pae1:uml::Class {name = an1,
268   ownedAttribute = opor1:uml::Port{name = sn,
269     isService=true, type=sin}}}};
270 when {
271   ParticipantsSoaMLrefs(po,pae,spart);
272   ParticipantsSoaMLrefs(p1,pael,spart1);
273   ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,sint,pro,cons,con,conn);
274 }
275 where {
276 PortSoaMLrefs(mec,inr,po,p1,co,pael,pael1,ser,req) or true;
277 }
278 }
279 relation PortSoaMLrefs {
280 an,cn,pn,bn,an1,sn,bn1,cn1:String;
281 checkonly domain bpmn mec:bpmn2::Message{name=bn};
282 checkonly domain bpmn inr:bpmn2::Interface{name=cn1,
283 operations=op:bpmn2::Operation{name = bn1,
284   inMessageRef=mec,outMessageRef=mec}};
285 checkonly domain bpmn po:bpmn2::Process{name = an,
286   flowElements=st:bpmn2::ServiceTask{name = cn, operationRef=op}};
287 checkonly domain bpmn p1:bpmn2::Process{name = an1,
288 flowElements=sr:bpmn2::Task {name = sn}};
289 checkonly domain bpmn co:bpmn2::Collaboration{
290 messageFlows=me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}};
291 enforce domain soaml pae:uml::Class {name = an,
292   ownedAttribute = opor:uml::Port{name = cn1}};
293 enforce domain soaml pae1:uml::Class {name = an1,
294   ownedAttribute = opor1:uml::Port{name = sn}};
295 enforce domain soaml ser:SoaML2::Service{base_Port = opor};
296 enforce domain soaml req:SoaML2::Request{base_Port = opor1};
297 }
298 /*top relation to define SoaML ServicesArchitecture from BPMN2 Collaboration*/
299 /*and generated SoaML elements */
300 top relation CollaborationToSoaMLServicesArchitecture{
301 pn,bn2:String;
302 sop:SoaML2::SoaMLPackage;
303 su:SoaML2::ServicesArchitecture;
304 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
305 rootElements=co:bpmn2::Collaboration{name = bn2}};
306 enforce domain soaml mo:uml::Model{name = pn,
307 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
308 packagedElement = uma:uml::Collaboration {name = bn2}}};
309   enforce domain soaml sup:SoaML2::SoaMLPackage{base_Package = pu};
310 when {
311 ModelToSoaMLModel(de,mo,sop);
312 }
313 where {
314 ServicesArchElementsToSoaMLServicesArchElements(de,mo,mo,su) or true;
315 }
316 }
317 relation ServicesArchElementsToSoaMLServicesArchElements{
318 pn,sn,bn,cn,an,an1,cn1,bn1,bn2:String;
319 sint:SoaML2::ServiceInterface;

```

```

320 pro:SoaML2::Provider;
321 cons:SoaML2::Consumer;
322 con:SoaML2::ServiceContract;
323 conn:SoaML2::ServiceChannel;
324 si:SoaML2::SoaMLPackage;
325 sop:SoaML2::SoaMLPackage;
326 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
327   rootElements=mec:bpmn2::Message{name=bn},
328   rootElements=incr:bpmn2::Interface{name=cn1,
329   operations=op:bpmn2::Operation{name = bn1,
330   inMessageRef=mec,outMessageRef=mec}},
331   rootElements=po:bpmn2::Process{name = an, flowElements=st:bpmn2::ServiceTask{name = cn,
332   operationRef=op}},
333   rootElements=p1:bpmn2::Process{name = an1, flowElements=sr:bpmn2::Task {name=sn}},
334 rootElements=co:bpmn2::Collaboration{name = bn2,
335   participants = par:bpmn2::Participant{name = an,
336   processRef = po:bpmn2::Process{name = an}},
337   participants = par1:bpmn2::Participant {name = an1,
338   processRef = p1:bpmn2::Process{name = an1}},
339   messageFlows = msf:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}}};
340 checkonly domain soaml mo:uml::Model{name = pn,
341 packagedElement = pa : uml::Package{name = 'Participants'+pn,
342 packagedElement = pae:uml::Class {name = an},
343 packagedElement = pae1:uml::Class {name = an1}},
344 packagedElement = pi : uml::Package {name = 'Services'+pn,
345   packagedElement = se :uml::Package{name = cn1+'Service',
346 packagedElement = sin:uml::Class{name=cn1},
347   packagedElement = col:uml::Collaboration{name=cn1,
348 ownedAttribute=ot:uml::Property{name='provider',type=sin,
349   aggregation=uml::AggregationKind::composite},
350   ownedAttribute=ot1:uml::Property{name='consumer',type=sin,
351   aggregation=uml::AggregationKind::composite}}}}};
352 enforce domain soaml mo:uml::Model{name = pn,
353 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
354 packagedElement = uma:uml::Collaboration {name = bn2,
355   ownedAttribute = oa:uml::Property{name = an, type = pae,
356   aggregation = uml::AggregationKind::composite},
357   ownedAttribute = oa1:uml::Property{name = an1, type = pae1,
358   aggregation = uml::AggregationKind::composite},
359   collaborationUse = cu:uml::CollaborationUse {name = cn1, type = col,
360   roleBinding = rb:uml::Dependency{supplier = oa:uml::Property{},
361   client = ot:uml::Property{},client = cu:uml::CollaborationUse{}},
362   roleBinding = rb1:uml::Dependency{supplier = oa1:uml::Property{type = pae1},
363   client = ot1:uml::Property{}, client = cu:uml::CollaborationUse{type = col}}}}}}};
364 enforce domain soaml su:SoaML2::ServicesArchitecture{base_Collaboration = uma};
365 when {
366   ElementsToSoamLElements(mec,inr,st,sr,co,se,si,sint,pro,cons,con,conn);
367 }
368 }
369 }

```

### C.3.3. Unidirectional option with simple UML Interfaces

The unidirectional option with simple UML Interfaces generates the same elements as presented in subsection C.2.3 preserving the names for the Interface, Operation and Messages identified for

the service provider, as presented in subsection C.3.1.

```

1 transformation bpmn2uml2soaml2 (bpmn:bpmn2, soaml:SoaML2) {
2   key uml::Interface {name};
3   key uml::Collaboration {name};
4   key uml::Class {name};
5   key uml::Package {name};
6   key uml::Property {name,type};
7   key uml::Property {name,default};
8   key uml::Operation {name};
9   key uml::Connector{end};
10  key uml::ConnectorEnd {role};
11  key uml::CollaborationUse{roleBinding};
12  key uml::Dependency {supplier,client};
13  key uml::Parameter {name};
14  key uml::Port {name};
15  /*top relation to define a SoaML model from the BPMN2 model of the BP*/
16  top relation ModelToSoaMLModel {
17    pn:String;
18    checkonly domain bpmn de:bpmn2::Definitions {name = pn};
19    enforce domain soaml mo:uml::Model{name = pn,
20      profileApplication = proa:uml::ProfileApplication{
21        appliedProfile = aproa:uml::Profile{
22          name = 'http://www.soaml.org/SoaML/1.0.0'}}};
23    enforce domain soaml sop:SoaML2::SoaMLPackage{
24      base_Package = mo};
25  }
26  /*top relation to define SoaML Participans from BPMN2 pools (process) of the BP*/
27  top relation ParticipantToSoaMLParticipant{
28    pn,pp:String;
29    sop:SoaML2::SoaMLPackage;
30    sa:SoaML2::Participant;
31    checkonly domain bpmn de:bpmn2::Definitions{name=pn,
32      rootElements=po:bpmn2::Process{name = pp}};
33    enforce domain soaml mo:uml::Model{name = pn,
34      packagedElement = pa:uml::Package{name = 'Participants'+pn,
35      packagedElement = uma:uml::Class {name = pp}}};
36    enforce domain soaml sap:SoaML2::SoaMLPackage{base_Package = pa};
37    when {
38      ModelToSoaMLModel(de,mo,sop);
39    }
40    where {
41      ParticipantsSoaMLrefs(po,uma,sa) or true;
42    }
43  }
44  relation ParticipantsSoaMLrefs{
45    pp:String;
46    checkonly domain bpmn po:bpmn2::Process{name = pp};
47    enforce domain soaml uma:uml::Class {name = pp};
48    enforce domain soaml sa:SoaML2::Participant{base_Class = uma};
49  }
50  /*top relation to define SoaML MessagesTypes from BPMN2 MessageFlows of the BP*/
51  top relation MessageToSoaMLMessage{
52    pn,bn:String;
53    sop:SoaML2::SoaMLPackage;
54    se:SoaML2::MessageType;

```

```

55     checkonly domain bpmn de:bpmn2::Definitions{name = pn,
56 rootElements=mec:bpmn2::Message{name=bn}};
57 enforce domain soaml mo:uml::Model{name = pn,
58 packagedElement = pe:uml::Package{name = 'Messages'+pn,
59 packagedElement = ume:uml::Class {name = bn+'Message'}}};
60 enforce domain soaml sep:SoaML2::SoaMLPackage{base_Package = pe};
61 when {
62 ModelToSoaMLModel(de,mo,sop);
63 }
64 where {
65     MessagesSoaMLrefs(mec,ume,se) or true;
66 }
67 }
68 relation MessagesSoaMLrefs{
69 bn:String;
70 checkonly domain bpmn mec:bpmn2::Message{name = bn};
71 checkonly domain soaml ume:uml::Class {name = bn+'Message'};
72 enforce domain soaml se:SoaML2::MessageType{base_Class = ume};
73 }
74 /*top relation to define SoaML Services from BPMN2 ServiceTasks, Tasks*/
75 /*and MessageFlows of the BP*/
76 top relation ServicesToSoaMLServices{
77 pn,sn,cn,bn,cn1,bn1:String;
78 pro:SoaML2::Provider;
79 con:SoaML2::ServiceContract;
80 conn:SoaML2::ServiceChannel;
81 si:SoaML2::SoaMLPackage;
82 sop:SoaML2::SoaMLPackage;
83 checkonly domain bpmn de:bpmn2::Definitions{name = pn,
84     rootElements=mec:bpmn2::Message{name=bn},
85     rootElements=incr:bpmn2::Interface{name=cn1,
86     operations=op:bpmn2::Operation{name = bn1,
87     inMessageRef=mec,outMessageRef=mec}},
88 rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn,
89 operationRef=op}},
90 rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
91 rootElements=co:bpmn2::Collaboration{messageFlows=
92 me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
93 enforce domain soaml mo:uml::Model{name = pn,
94 packagedElement = pi:uml::Package{name = 'Services'+pn,
95     packagedElement = se:uml::Package {name = cn1+'Service'}}};
96 enforce domain soaml sip:SoaML2::SoaMLPackage{base_Package = pi};
97 when {
98 ModelToSoaMLModel(de,mo,sop);
99 }
100 where {
101     ElementsToSoaMLElements(mec,incr,st,sr,co,se,si,pro,con,conn) or true;
102 }
103 }
104 relation ElementsToSoaMLElements{
105 cn,sn,bn,cn1,bn1:String;
106 checkonly domain bpmn mec:bpmn2::Message{name=bn};
107     checkonly domain bpmn incr:bpmn2::Interface{name=cn1,
108     operations=op:bpmn2::Operation{name = bn1,
109     inMessageRef=mec,outMessageRef=mec}};
110 checkonly domain bpmn st:bpmn2::ServiceTask{name = cn};

```

```

111 checkonly domain bpmn sr:bpmn2::Task{name = sn};
112 checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
113 me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}};
114 enforce domain soaml se:uml::Package{name = cn1+'Service',
115 packagedElement = it:uml::Interface{name= cn1,
116   ownedOperation=owp:uml::Operation{name = bn1,
117   visibility=uml::VisibilityKind::public,
118     ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1In',
119     direction=uml::ParameterDirectionKind::inout},
120     ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
121     direction=uml::ParameterDirectionKind::return}}},
122 packagedElement = col:uml::Collaboration{ name=cn1,
123 ownedAttribute=ot:uml::Property{name='provider',type=it,
124   aggregation=uml::AggregationKind::composite},
125 ownedAttribute=ot1:uml::Property{name='consumer', default=sn,
126   aggregation=uml::AggregationKind::composite},
127 ownedConnector = oc:uml::Connector{
128   end = en:uml::ConnectorEnd {role = ot,isUnique=false},
129   end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}};
130 enforce domain soaml si:SoaML2::SoaMLPackage{base_Package = se};
131 enforce domain soaml pro:SoaML2::Provider{base_Interface = it};
132 enforce domain soaml con:SoaML2::ServiceContract{base_Collaboration = col};
133 enforce domain soaml conn:SoaML2::ServiceChannel {base_Connector = oc:uml::Connector{}};
134 }
135 /*top relation to update SoaML Services adding MessageTypes as the*/
136 /*type of the parameters */
137 top relation MessageToServiceSoaMLMessage{
138   pn,bn,cn,sn,cn1,bn1:String;
139   mes:SoaML2::MessageType;
140 si:SoaML2::SoaMLPackage;
141 pro:SoaML2::Provider;
142 con:SoaML2::ServiceContract;
143 conn:SoaML2::ServiceChannel;
144   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
145 rootElements=mec:bpmn2::Message{name=bn},
146   rootElements=incr:bpmn2::Interface{name=cn1,
147   operations=op:bpmn2::Operation{name = bn1,
148   inMessageRef=mec,outMessageRef=mec}},
149 rootElements=po:bpmn2::Process{flowElements=st:bpmn2::ServiceTask{name = cn,
150 operationRef=op}},
151 rootElements=p1:bpmn2::Process{flowElements=sr:bpmn2::Task{name = sn}},
152 rootElements=co:bpmn2::Collaboration{messageFlows=
153 me:bpmn2::MessageFlow{targetRef=st, sourceRef=sr}}};
154 checkonly domain soaml mo:uml::Model{name = pn,
155 packagedElement = pe : uml::Package{name = 'Messages'+pn,
156 packagedElement = mae:uml::Class {name = bn+'Message'}},
157 packagedElement = pi : uml::Package {name = 'Services'+pn,
158   packagedElement = se :uml::Package{name = cn1+'Service',
159   packagedElement = it:uml::Interface{name=cn1,
160   ownedOperation=owp:uml::Operation{name = bn1,
161   visibility=uml::VisibilityKind::public,
162   ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1In',
163   direction=uml::ParameterDirectionKind::inout},
164   ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
165   direction=uml::ParameterDirectionKind::return}}}}}};
166 enforce domain soaml mo:uml::Model{name = pn,

```

```

167     packagedElement = pi : uml::Package {name = 'Services'+pn,
168     packagedElement = se :uml::Package{name = cn1+'Service',
169 packagedElement = it:uml::Interface{name=cn1,
170     ownedOperation=owp:uml::Operation{name = bn1,
171     visibility=uml::VisibilityKind::public,
172 ownedParameter = owpa:uml::Parameter{name=bn1+'Parameter1In',
173 direction=uml::ParameterDirectionKind::inout, type = mae},
174 ownedParameter = owpe:uml::Parameter{name=bn1+'Return',
175 direction=uml::ParameterDirectionKind::return, type = mae}}}}}}};
176 when {
177     MessagesSoaMLrefs(mec,mae,mes);
178     ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,pro,con,conn);
179 }
180 }
181 /*top relation to update SoaML Participants adding Ports*/
182 /*corresponding to generated services */
183     top relation PortToSoaMLPort{
184     pn,bn,an,cn,an1,sn,cn1,bn1:String;
185     spart,spart1:SoaML2::Participant;
186     ser:SoaML2::Service;
187     req:SoaML2::Request;
188     pro:SoaML2::Provider;
189     con:SoaML2::ServiceContract;
190     conn:SoaML2::ServiceChannel;
191     si:SoaML2::SoaMLPackage;
192     checkonly domain bpmn de:bpmn2::Definitions{name = pn,
193     rootElements=mec:bpmn2::Message{name=bn},
194     rootElements=inr:bpmn2::Interface{name=cn1,
195     operations=op:bpmn2::Operation{name = bn1,
196     inMessageRef=mec,outMessageRef=mec}},
197     rootElements=po:bpmn2::Process{name = an, flowElements=st:bpmn2::ServiceTask{name=cn,
198     operationRef=op}},
199     rootElements=p1:bpmn2::Process{name= an1, flowElements=sr:bpmn2::Task{name=sn}},
200     rootElements=co:bpmn2::Collaboration{
201     messageFlows = msf:bpmn2::MessageFlow{targetRef = st,sourceRef=sr}}};
202     checkonly domain soaml mo:uml::Model{name = pn,
203     packagedElement = pa : uml::Package{name = 'Participants'+pn,
204     packagedElement = pae:uml::Class {name = an},
205     packagedElement = pae1:uml::Class {name = an1}},
206     packagedElement = pi : uml::Package {name = 'Services'+pn,
207     packagedElement = se :uml::Package{name = cn1+'Service',
208     packagedElement = it:uml::Interface{name=cn1}}}}};
209     enforce domain soaml mo:uml::Model{name = pn,
210     packagedElement = pa : uml::Package{name = 'Participants'+pn,
211     packagedElement = pae:uml::Class {name = an,
212     ownedAttribute = opor:uml::Port{name = cn1, isService=true, type=it}},
213     packagedElement = pae1:uml::Class {name = an1,
214     ownedAttribute = opor1:uml::Port{name = sn, isService=true, type=it,
215     isConjugated=true}}}}};
216     when {
217     ParticipantsSoaMLrefs(po,pae,spart);
218     ParticipantsSoaMLrefs(p1,pae1,spart1);
219     ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,pro,con,conn);
220     }
221     where {
222     PortSoaMLrefs(mec,inr,po,p1,co,pae,pae1,ser,req) or true;

```



```

223 }
224 }
225 relation PortSoaMLrefs {
226   an,cn,bn,an1,sn,cn1,bn1,bn2:String;
227   checkonly domain bpmn mec:bpmn2::Message{name=bn};
228     checkonly domain bpmn inr:bpmn2::Interface{name=cn1,
229       operations=op:bpmn2::Operation{name = bn1,
230         inMessageRef=mec,outMessageRef=mec}};
231   checkonly domain bpmn po:bpmn2::Process{name = an,
232     flowElements=st:bpmn2::ServiceTask{name = cn}};
233   checkonly domain bpmn p1:bpmn2::Process{name = an1,
234     flowElements=sr:bpmn2::Task {name = sn}};
235     checkonly domain bpmn co:bpmn2::Collaboration{messageFlows=
236   me:bpmn2::MessageFlow{targetRef=st,sourceRef=sr}};
237   enforce domain soaml pae:uml::Class {name = an,
238     ownedAttribute = opor:uml::Port{name = cn1}};
239   enforce domain soaml pae1:uml::Class {name = an1,
240     ownedAttribute = opor1:uml::Port{name = sn}};
241   enforce domain soaml ser:SoaML2::Service{base_Port = opor};
242   enforce domain soaml req:SoaML2::Request{base_Port = opor1};
243 }
244 /*top relation to define SoaML ServicesArchitecture from BPMN2 Collaboration*/
245 /*and generated SoaML elements */
246 top relation CollaborationToSoaMLServicesArchitecture{
247   pn,bn2:String;
248   sop:SoaML2::SoaMLPackage;
249   su:SoaML2::ServicesArchitecture;
250   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
251     rootElements=co:bpmn2::Collaboration{name = bn2}};
252   enforce domain soaml mo:uml::Model{name = pn,
253     packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
254     packagedElement = uma:uml::Collaboration {name = bn2}}};
255     enforce domain soaml sup:SoaML2::SoaMLPackage{base_Package = pu};
256   when {
257     ModelToSoaMLModel(de,mo,sop);
258   }
259   where {
260     ServicesArchElementsToSoaMLServicesArchElements(de,mo,mo,su)or true;
261   }
262 }
263 relation ServicesArchElementsToSoaMLServicesArchElements{
264   pn,sn,bn,cn,an,an1,cn1,bn1,bn2:String;
265   pro:SoaML2::Provider;
266   con:SoaML2::ServiceContract;
267   conn:SoaML2::ServiceChannel;
268   si:SoaML2::SoaMLPackage;
269   checkonly domain bpmn de:bpmn2::Definitions{name = pn,
270     rootElements=mec:bpmn2::Message{name=bn},
271     rootElements=inr:bpmn2::Interface{name=cn1,
272       operations=op:bpmn2::Operation{name = bn1,
273         inMessageRef=mec,outMessageRef=mec}}},
274   rootElements=po:bpmn2::Process{name=an,flowElements=st:bpmn2::ServiceTask{name=cn,
275     operationRef=op}},
276     rootElements=p1:bpmn2::Process{name = an1, flowElements=sr:bpmn2::Task{name=sn}},
277   rootElements=co:bpmn2::Collaboration{name = bn2,
278     participants = par:bpmn2::Participant{name = an, processRef = po},

```

```

279     participants = par1:bpnm2::Participant {name = an1, processRef = p1},
280     messageFlows = msf:bpnm2::MessageFlow{targetRef = st, sourceRef = sr}}};
281 checkonly domain soaml mo:uml::Model{name = pn,
282 packagedElement = pa : uml::Package{name = 'Participants'+pn,
283 packagedElement = pae:uml::Class {name = an},
284 packagedElement = pae1:uml::Class {name = an1}},
285 packagedElement = pi : uml::Package {name = 'Services'+pn,
286     packagedElement = se :uml::Package{name = cn1+'Service',
287 packagedElement = it:uml::Interface{name=cn1},
288     packagedElement = col:uml::Collaboration{name=cn1,
289     ownedAttribute=ot:uml::Property{name='provider',type=it,
290     aggregation=uml::AggregationKind::composite},
291     ownedAttribute=ot1:uml::Property{name='consumer', default=sn,
292     aggregation=uml::AggregationKind::composite},
293     ownedConnector = oc:uml::Connector{
294     end = en:uml::ConnectorEnd {role = ot,isUnique=false},
295     end = en1:uml::ConnectorEnd {role = ot1,isUnique=false}}}}}}};
296 enforce domain soaml mo:uml::Model{name = pn,
297 packagedElement = pu:uml::Package{name = 'ServicesArchitecture'+pn,
298 packagedElement = uma:uml::Collaboration {name = bn2,
299     ownedAttribute = oa:uml::Property{name = an, type = pae,
300     aggregation = uml::AggregationKind::composite},
301     ownedAttribute = oa1:uml::Property{name = an1, type = pae1,
302     aggregation = uml::AggregationKind::composite},
303     collaborationUse = cu:uml::CollaborationUse {name = cn1, type = col,
304     roleBinding = rb:uml::Dependency{supplier = oa:uml::Property{},
305     client = ot:uml::Property{},
306     client = cu:uml::CollaborationUse{}},
307     roleBinding = rb1:uml::Dependency{supplier = oa1:uml::Property{type = pae1},
308     client = ot1:uml::Property{},
309     client = cu:uml::CollaborationUse{type = col}}}}}}};
310 enforce domain soaml su:SoaML2::ServicesArchitecture{base_Collaboration = uma};
311 when {
312     ElementsToSoaMLElements(mec,inr,st,sr,co,se,si,pro,con,conn);
313 }
314 }
315 }

```

## Appendix D.

# Experimental material of the QVT transformations validation experiment

### D.1. Overview

This Appendix presents the experimental material for the QVT transformations experiments as designed to be carried out in the experiment (in Spanish only) for Group A, the Group B used the same material but in a different order. The material presented here is the original one specified in a text editor that had to be adapted for visualization in the WebGen application, but without altering the contents.

### D.2. Experimental materials

#### D.2.1. Tutorial

Each subject received the address to view or download the tutorial in pdf format from googledocs<sup>1</sup> after he/she has agreed to perform the experiment. For each notation its main elements were described showing the corresponding icons and definitions: for BPMN2 several core elements for Flow objects, Connection objects, Swimlanes and Artifacts, with focus on the ones used in the experiment, and for SoaML the different type of diagrams and communication pattern for designing bidirectional or unidirectional services were described.

An example exercise for each of the two parts of the experiment was also presented, for the BP model Buyer-Reseller showing the corresponding SoaML diagrams, the textual correspondence rules, and the true or false questions for the service design, explaining the answers but without actually answering them so the subjects had no guide about the options. The tutorial was ten pages long from which three corresponded to BPMN2, three to SoaML and four for the example, and which took approximately between twenty and thirty minutes to read and understand it.

---

<sup>1</sup><https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B2uspN0K-noVMGZjNjLzYjYtNjUwNi00MWE4LTg1MDEtNjk2NzIwMmU1M2Q5&hl=es>

**Nodos:** son usados para controlar la divergencia / convergencia del flujo de control del proceso (XOR, OR, AND). Ejemplos nodos (entre otros):

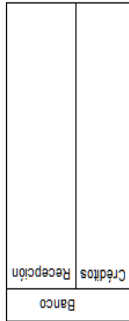


**Swimlanes:**

**Pool:** representan participantes en el proceso de negocio, pueden ser una entidad de negocio (organización como Hospital, Universidad, Banco) o un rol (comprador, vendedor). Ejemplo:



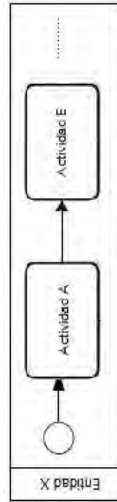
**Lane:** representan sub-particiones de los pools, pueden ser roles en la organización (Administrativo, Secretaria, Recepcionista), secciones (Marketing, Ventas, Contabilidad), sistemas, etc. Ejemplo:



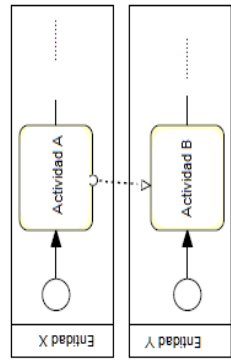
**Objetos de conexión:**

Permiten conectar los objetos de flujo entre sí para crear la estructura del PN. Tienen un único objeto origen y un único objeto destino.

**Secuencia:** muestra el orden en que se llevarán a cabo las tareas del PN contenido en una única entidad (pool) y no se pueden usar para conectar dos entidades (distintos pools). Ejemplo entre dos actividades:



**Mensaje:** muestra la interacción entre dos entidades (distintos pools) y no se pueden usar dentro de una única entidad (pool). Ejemplo mensajes entre dos actividades en distintos pools:



**Tutorial Experimento MINERVA**

Este tutorial se compone de tres partes: en la **primera** se describen los elementos principales de la **notación BPMN2 para modelado de Procesos de Negocio**, en la **segunda** se describen los elementos principales de la **notación SoaML (perfil UML) para modelado de servicios** y en la **tercera** se muestran **dos ejercicios de ejemplo**: uno para la **primer parte del experimento** y otro para la **segunda parte del experimento**. Por favor, lea atentamente este tutorial antes de realizar el experimento.

**PARTE 1: notación BPMN2**

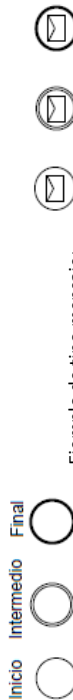
Un proceso de negocio (PN) consiste en un conjunto de actividades que son realizadas coordinadamente en un ambiente organizacional y tecnológico, en pos de un objetivo del negocio. Cada proceso de negocio es ejecutado en una organización pero puede interactuar con procesos de negocio en otras organizaciones. Un modelo de proceso de negocio es una representación de un proceso de negocio de una/s organización/es, BPMN2 permite especificar modelos de procesos de negocio mediante un conjunto de elementos definidos.

**Categorías y elementos principales en un modelo de PN:**

Objetos de Flujo	Objetos de Conexión	Swimlane	Artefactos
<p>Eventos</p> <p>Actividades</p> <p>Nodos de Decisión/union</p>	<p>Flujos de Secuencia</p> <p>Flujos de Mensaje</p> <p>Asociación</p>	<p>Pool (Entidad/Rol)</p> <p>Lane (participante dentro de un pool)</p>	<p>Objeto de datos</p> <p>Anotación de texto</p> <p>Grupo</p>

**Objetos de flujo:**

**Eventos:** algo que sucede en el curso del PN que afecta el flujo del proceso. Los eventos pueden ser de distintos tipos clasificados en eventos de inicio, intermedios y fin:



Ejemplo de tipo mensaje:

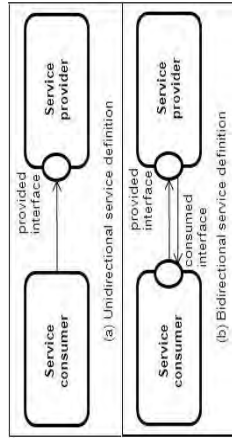


**Actividades:** término genérico para describir una unidad de trabajo, que puede ser atómica o compuesta. Ejemplos actividades (entre otras):



**PARTE 2: notación SoaML**

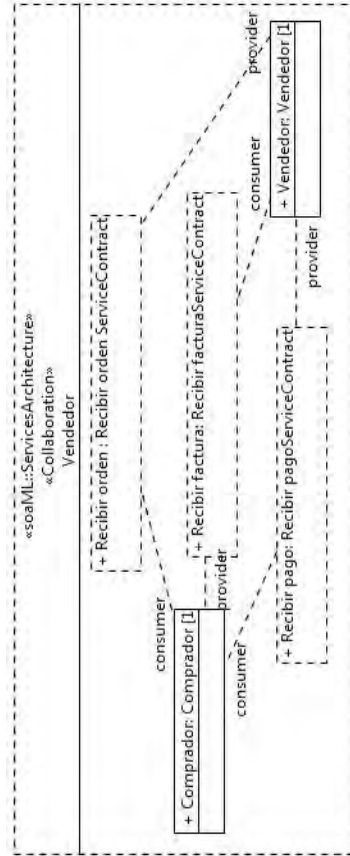
Un servicio (de software) es un recurso que permite el acceso a una o más capacidades provistas por un "proveedor" y usadas por un "consumidor". El acceso es provisto mediante una interface definida y ejecutado según restricciones y políticas especificadas en la descripción del servicio, los que son provistos y consumidos en Puertos UML de los participantes (proveedor, consumidor). Los consumidores pueden o no ser conocidos por el proveedor. La interacción con un servicio se describe mediante patrones de intercambio de mensajes (MEPs); SoaML provee dos tipos de comunicación: bidireccional y unidireccional. En el bidireccional el proveedor ofrece una interface para ser invocada por el consumidor y el consumidor ofrece una interface para que el proveedor invoque como respuesta en la interacción. En el unidireccional solo el proveedor ofrece una interface para ser invocada por el consumidor. Ejemplo:



Un modelo de servicios se compone de varios diagramas que permiten distintas vistas del diseño definido: Arquitectura de servicios, Interfaces, Contratos de servicios, Participantes y puertos de servicios, Tipos de mensajes y Capacidades. **En el experimento usaremos únicamente los cuatro primeros y la comunicación bidireccional entre servicios.**

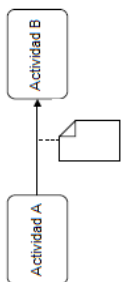
**Arquitectura de Servicios:**

La Arquitectura de Servicios define los participantes y los servicios (contratos de) provistos y consumidos por cada uno, indicando estos roles en las asociaciones entre participantes y contratos de servicios modelados. Es una colaboración UML estereotipada con el estereotipo "ServiceArchitecture". Ejemplo:



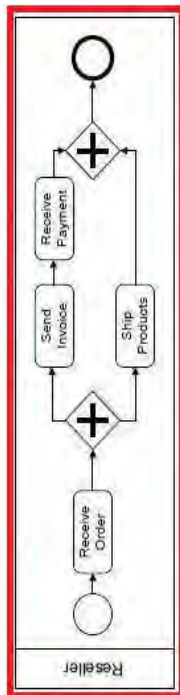
Los servicios entre los distintos participantes se pueden identificar de varias formas distintas para soportar actividades, sub-procesos o procesos de negocio enteros. Para esto hay que identificar las partes del proceso de negocio que pueden ser automatizadas y diseñar los servicios que las implementen.

**Asociación:** permite asociar objetos de datos con objetos de flujo o conexión. Ejemplo asociar un documento a una secuencia indicando el envío de datos asociado:

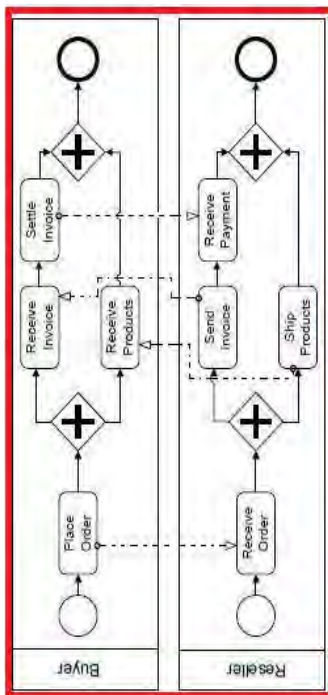


**Ejemplos de tipos de Procesos de negocio:**

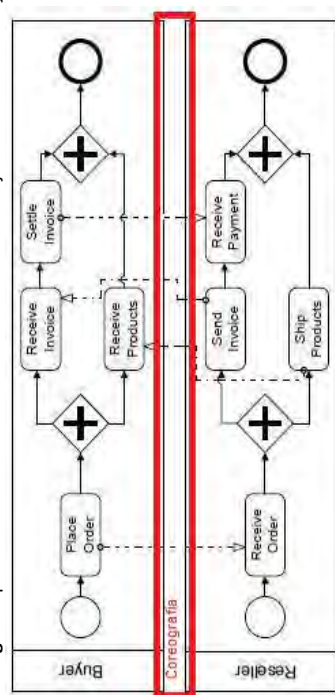
**Orquestación:** un proceso interno a una única organización que controla su flujo de ejecución. Ejemplo:



**Colaboración:** un proceso en el que participan dos o más organizaciones donde el flujo es acordado y su control compartido por los participantes, mediante el intercambio de mensajes. Ejemplo:

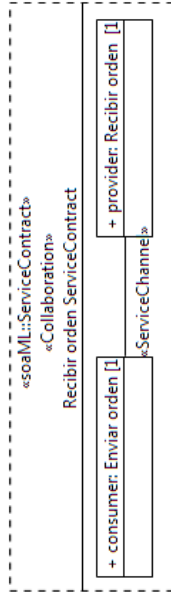


**Coreografía:** parte de una colaboración con foco en los mensajes intercambiados. Ejemplo:



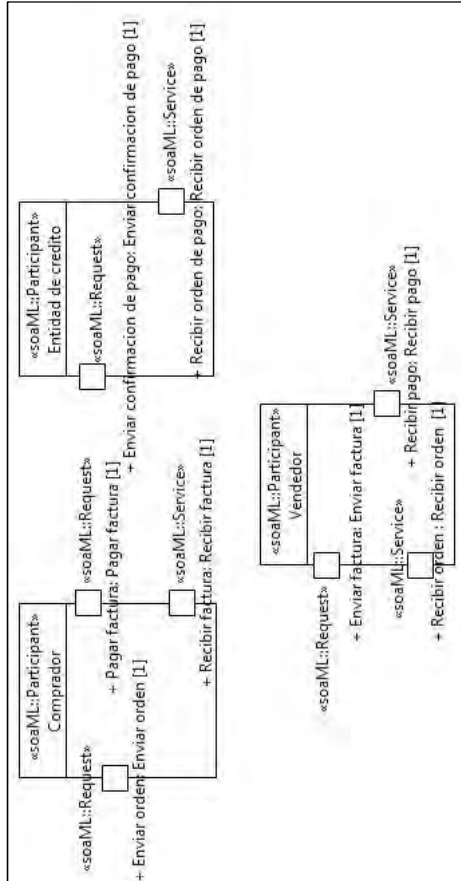
**Contratos de servicios:**

Un **Contrato de servicios especifica todos los elementos necesarios para definir el servicio**, integrando las interfaces definidas (con operaciones, parámetros y sus tipos), pre y post condiciones para el uso de las operaciones, roles consumidor y proveedor en base a las interfaces definidas, y definiendo una coreografía de comportamiento (si es necesario) para la interacción. Un **contrato de servicios es una colaboración UML con el estereotipo "ServiceContract"**, donde se definen los roles "consumer" y "provider" que son tipados con las interfaces definidas previamente para los mismos. Ejemplo para servicio bidireccional con interfaces simple UML:



**Participantes y servicios asociados:**

Los **participantes se corresponden con las entidades (organizaciones, sistemas, roles) que participan en la interacción mediante servicios** definida. Los **participantes proveen servicios en Puertos UML con el estereotipo "Service"** y **requieren servicios en Puertos UML con el estereotipo "Request"**. Los **puertos son tipados con las interfaces definidas según los roles consumidor y proveedor especificados en el contrato**. Ejemplo de participantes con servicios definidos con comunicación bidireccional:

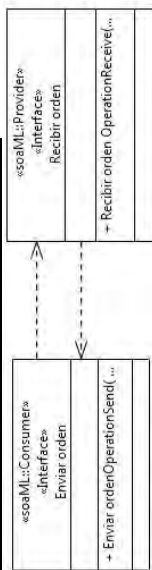


En el caso **unidireccional el puerto "Request" del consumidor será tipado con el conjugado de la interface del proveedor, para indicar su uso: ~InterfaceProveedor**. En el caso de **definición con ServiceInterface ambos puertos "Request" del consumidor y "Service" del proveedor serán tipados con la ServiceInterface**, donde la interface que "usa" será la del consumidor y la que "realiza" será la del proveedor. Estos casos no se verán en el experimento solo se usará la **opción de diseño bidireccional con interfaces simples UML**.

**Interfaces de servicios:**

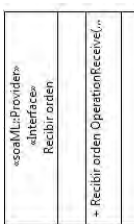
Las **Interfaces de servicios** son parte de la especificación de los servicios y definen las **operaciones, parámetros y tipos necesarios para interactuar con el servicio**. Pueden **modelarse como Interfaces simples UML o como Interfaces de Servicios con el estereotipo "ServiceInterface" de SoaML** que realizan y usan Interfaces simples UML, agregando otros elementos. Los **servicios se definen siempre desde el punto de vista del proveedor** que es quién provee el servicio a consumir. En el **experimento usaremos únicamente Interfaces simples UML**. Ejemplos:

**Interfaces simples UML comunicación bidireccional:**



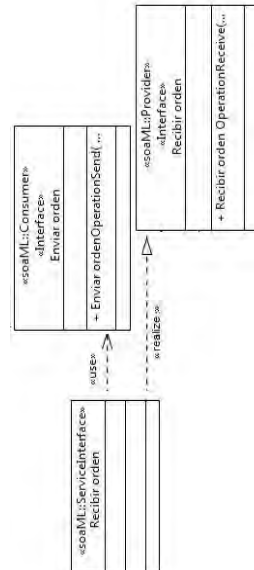
En este servicio el **consumidor Interface "Enviar Orden"** invoca al **proveedor Interface "Recibir orden"** en la operación **"Recibir orden OperationReceive(...)"** que éste provee, **para iniciar la interacción definida para el servicio "Recibir orden"**. El **proveedor Interface "Recibir orden"** contesta invocando a la operación **"Enviar orden Operation Send (...)"** de la **Interface del consumidor "Enviar Orden"**. El par de invocaciones es en general asincrónico, y cada una puede a su vez ser sincrónica o asincrónica.

**Interfaces simples UML comunicación unidireccional:**



En este servicio se cuenta **únicamente con la Interface del proveedor "Recibir orden" que debe ser invocado en la operación "Recibir orden Operation Receive(...)" para iniciar la interacción con el servicio**. Esta interacción puede ser sincrónica o asincrónica.

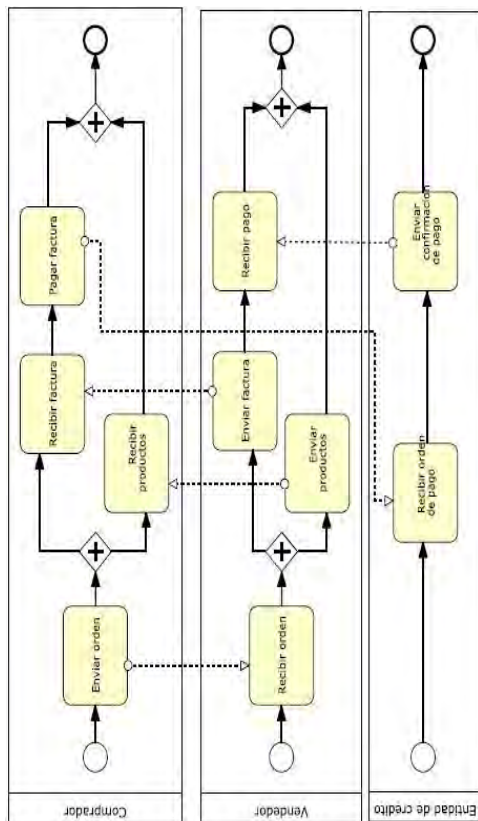
**Interfaces de servicios (ServiceInterface) comunicación bidireccional:**



En este servicio se **modela el uso y la realización de Interfaces simples UML para el servicio por medio de la definición de una ServiceInterface**, que **"usa" la Interface del consumidor "Enviar orden" y "realiza" la interface del proveedor "Recibir orden"**. La interacción con el servicio es bidireccional idem a la primer opción presentada.

**PARTE 3: Ejemplo de ejercicios del experimento**

Proceso de Negocio "Venta de productos" de un Vendedor:



**Descripción del Proceso de Negocio:**

El comprador envía una orden de compra al vendedor mediante la página Web de venta de productos que este provee. El vendedor recibe la orden y dispara dos eventos en paralelo:

1- emite y envía la factura al comprador

2 - emite el envío de los productos solicitados

El comprador recibe la factura enviada por el vendedor y envía la orden de pago correspondiente a la Entidad de crédito, recibiendo los productos solicitados.

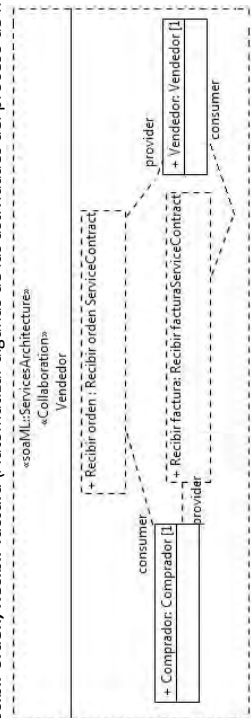
La entidad de crédito realiza el pago indicado en la orden de pago y envía la confirmación del pago al Vendedor. No se modelan excepciones como pago no autorizado o problemas con el envío de productos.

**Experimento parte 1:**

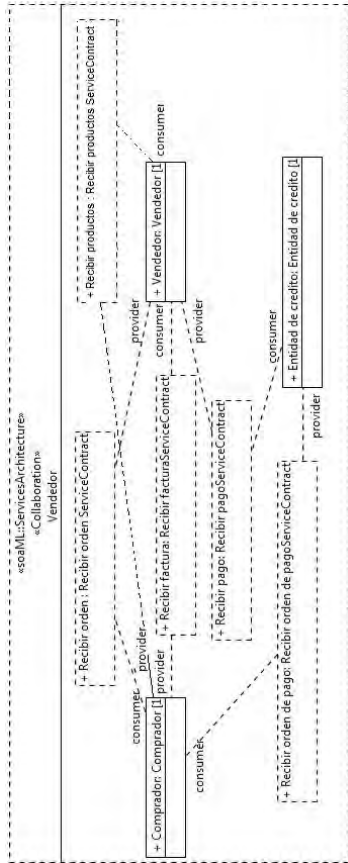
1 - ejercicio de diagramas SoaML para implementar el proceso de negocio con distintas opciones de servicios para elegir. Ejemplo Arquitectura de Servicios, elegir una de las 4 opciones brindadas a, b, c, d.

**a) Participantes: Comprador, Vendedor**

**Servicios: Recibir orden, Recibir factura** (Automatizar algunas de las actividades del proceso de negocio)

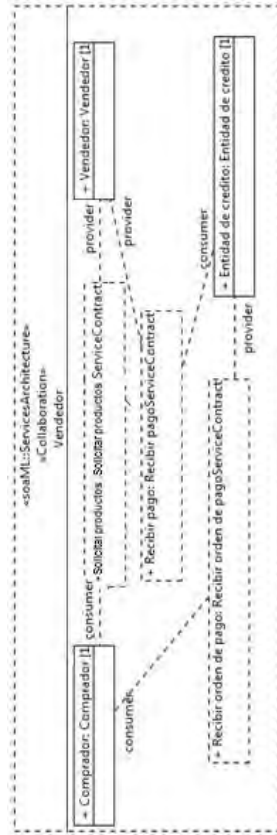


**b) Participantes: Comprador, Vendedor, Entidad de crédito**  
**Servicios: Recibir orden, Recibir factura, Recibir pago, Recibir productos, Recibir orden de pago.** (Automatizar todas las actividades del proceso de negocio, incluso las manuales)



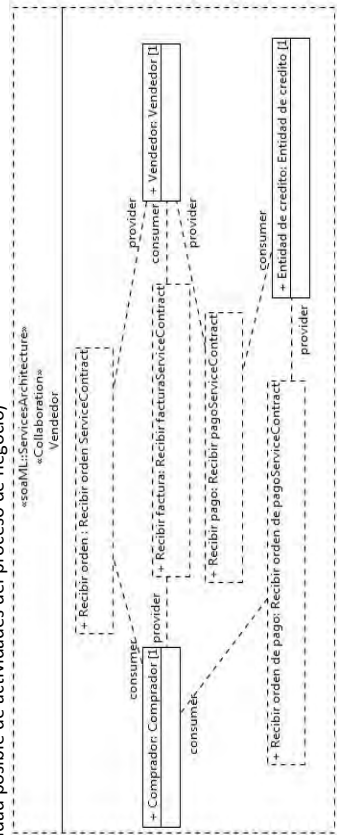
**c) Participantes: Comprador, Vendedor, Entidad de crédito**

**Servicios: Solicitar productos, Recibir orden de pago, Recibir pago.** (Automatizar algunas actividades del proceso de negocio y combinar otras en un servicio que haga las tareas de las actividades combinadas)



**d) Participantes: Comprador, Vendedor, Entidad de crédito**

**Servicios: Recibir orden, Recibir factura, Recibir orden de pago, Recibir pago.** (Automatizar la mayor cantidad posible de actividades del proceso de negocio)



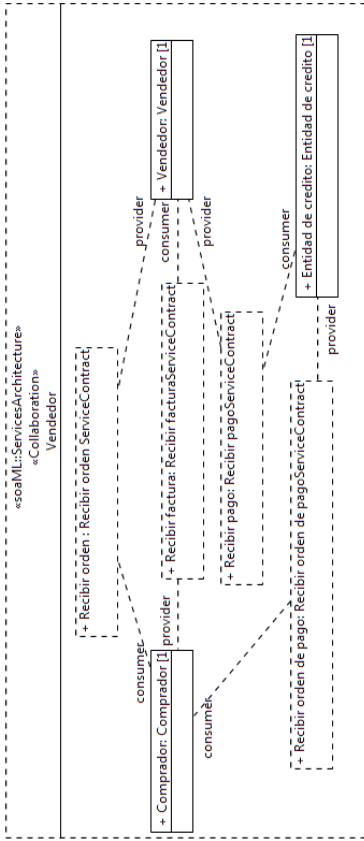
2 – ejercicio de reglas de definición de servicios desde procesos de negocio con distintas opciones de servicios para elegir. Elija una de las 4 opciones brindadas a, b, c, d.

La Arquitectura de Servicios se corresponde con:

- a) el proceso colaborativo “Venta de productos” e incluye:
- los participantes: **Comprador, Vendedor, Entidad de crédito**
- los servicios que se corresponden con actividades y combinación conceptual de actividades involucradas en un flujo de mensajes entre dos procesos distintos:
  - **Solicitar productos** (proveedor Vendedor, consumidor Comprador)
  - **Recibir orden de pago** (proveedor Comprador, consumidor Entidad de crédito)
  - **Recibir pago** (proveedor Vendedor, consumidor Entidad de crédito)
- b) el proceso colaborativo “Venta de productos” e incluye:
- los participantes: **Comprador, Vendedor y Entidad de crédito**
- los servicios que se corresponden con las actividades involucradas en un flujo de mensajes entre los dos procesos:
  - **Recibir orden** (proveedor Vendedor, consumidor Comprador)
  - **Recibir factura** (proveedor Comprador, consumidor Vendedor)
  - **Recibir productos** (proveedor Comprador, consumidor Vendedor)
  - **Recibir orden de pago** (proveedor Entidad de crédito, consumidor Comprador)
  - **Recibir pago** (proveedor Vendedor, consumidor Entidad de crédito)
- c) el proceso colaborativo “Venta de productos” e incluye:
- los participantes: **Comprador, Vendedor y Entidad de crédito**
- los servicios que se corresponden con las actividades involucradas en un flujo de mensaje entre los dos procesos:
  - **Recibir orden** (proveedor Vendedor, consumidor Comprador)
  - **Recibir factura** (proveedor Comprador, consumidor Vendedor)
  - **Recibir orden de pago** (proveedor Entidad de crédito, consumidor Comprador)
  - **Recibir pago** (proveedor Vendedor, consumidor Entidad de crédito)
- d) los procesos **Comprador y Vendedor** integrantes del proceso colaborativo “Venta de productos” e incluye:
- los participantes: **Comprador, Vendedor**
- los servicios que se corresponden con las actividades involucradas en un flujo de mensaje entre dos procesos distintos:
  - **Recibir orden** (proveedor Vendedor, consumidor Comprador)
  - **Recibir factura** (proveedor Comprador, consumidor Vendedor)

Experimento parte 2:

1 – ejercicio de diagramas SoaML identificando distintos elementos en el diagrama presentado para contestar Verdadero/Falso. Conteste las 6 preguntas correspondientes al diagrama.



- 1.1) El participante **Vendedor** provee los servicios “**Recibir orden**”, “**Recibir factura**”, “**Recibir pago**”, “**Recibir orden de pago**”.
- 1.2) El rol **provider** en el contrato de servicio del servicio “**Recibir orden de pago**” es jugado por el participante **Entidad de crédito**.
- 1.3) El participante **Paciente** juega el rol **consumer** definido en el contrato de servicio del servicio “**Recibir factura**”.
- 1.4) El contrato de servicio para el servicio “**Recibir orden**” define la interacción de los participantes **Comprador y Vendedor** con el servicio con los roles **consumer y provider** respectivamente.
- 1.5) Los roles definidos para el servicio “**Recibir factura**” **consumer y provider** son jugados por los participantes **Comprador y Vendedor** respectivamente.
- 1.6) El participante **Entidad de crédito** interactúa con el participante **Comprador** en el servicio “**Recibir pago**” con el rol **consumer**.

2 – ejercicio de diagramas SoaML para indicar la complejidad que considera que presenta el diagrama brindado en escala del 1 al 5 definida. Valore la complejidad del modelo de Arquitectura de Servicios SoaML presentado en el ejercicio 1:

- 1 – Muy simple
- 2 – Algo simple
- 3 – Normal
- 4 – Algo complejo
- 5 – Muy complejo



### **D.2.2. Part 1**

Part 1 of the experiment which evaluates the Suitability of the QVT transformations, consisted in performing four exercises for each of the two BP models provided, where in each exercise a different SoaML diagram for the design of services to realize the BP are given: Services Architecture, Services Interfaces, Services Contracts and Participants & Services. For each exercise the order in which the diagrams were presented was the same as described previously, simulating a top - down design approach where first the services are identified and then they specified. The four SoaML diagrams provided correspond to a different design solution, that traverses from the first diagram to the last one, each one corresponding to:

1. the design generated from the QVT transformations
2. a design in where related activities are combined to define services
3. a design where a service is derived from each activity in a message flow
4. a design where services are defined only for a few arbitrary selected activities

Although each design option was maintained throughout the four SoaML diagrams, the order in which the options were presented for each diagram was changed, that is if the order in the first SoaML diagram was the one enumerated above 1-2-3-4, for the second diagram the order of the options could be 3-4-1-2, for the third diagram the order of the options could be 4-2-3-1 and the order for the fourth diagram could be 2-3-1-4, and referring to elements in each of the diagrams.

Each exercise consisted also in two parts: one where the design options were provided only with the SoaML diagrams and one had to be selected, and another where the design options were provided as textual correspondence rules from which one had also to be selected. The order of the diagrams - rules was different for each BP model for each defined group A and B, as described before.

Cuestionario Inicial:

- 1) Indique su formación
  - a) Estudiante de grado en Informática (Computación)
  - b) Estudiante de maestría (o posgrado similar) en Informática
  - c) Estudiante de doctorado en Informática (Computación)
  - d) Profesor (Académico) en Informática (Computación)
  - e) Profesional de Informática (Computación)

2) Indique su país de origen

---

3) Indique su filiación (Universidad XX, Empresa YY)

---

4) Indique su conocimiento del lenguaje UML
 

- a) Muy bajo
- b) Bajo
- c) Medio
- d) Alto
- e) Medio alto

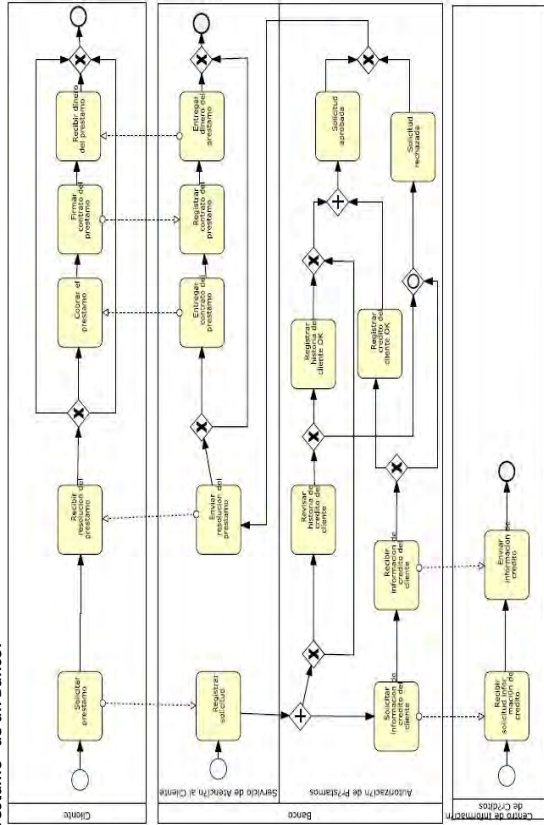
5) Indique su conocimiento del lenguaje SoaML
 

- a) Muy bajo
- b) Bajo
- c) Medio
- d) Alto
- e) Medio alto

6) Indique su conocimiento del lenguaje BPMN2
 

- a) Muy bajo
- b) Bajo
- c) Medio
- d) Alto
- e) Medio alto

Sección 1 – Planteamiento del problema: descripción del Proceso de negocio colaborativo “Otorgar Préstamo” de un Banco.



**Descripción del proceso de negocio “Otorgar Préstamo”**

El Cliente solicita un préstamo en el formulario de “Solicitud de Préstamo” del sitio web del Banco, para lo cual ingresa con su usuario y contraseña.

El Banco registra la solicitud ingresada y dispara dos eventos en paralelo:

- 1 - solicitar información de la historia de crédito del Cliente en el Centro de Información de créditos
- 2 – si el monto es >=2000 euros se envía la solicitud ingresada para que la historia de crédito del Cliente con el Banco sea revisada por personal de la sección Autorización de Préstamos.

Si el informe del Centro de Información de Créditos es favorable y el informe de la sección Autorización de Préstamos es favorable o el monto es < 2000 el crédito es aprobada.

Si el informe del Centro de Información de Créditos o el Informe de la sección Autorización de Préstamos no es favorable entonces la solicitud de crédito es rechazada.

En cualquier caso se informa al cliente de la resolución mediante el envío de un mail a su casilla registrada en el Banco y/o un sms si está registrado en este servicio, o una llamada telefónica si no se cumple ninguna de las anteriores.

Si la solicitud fue aprobada el Cliente concurre al Banco a firmar el contrato del préstamo otorgado que se registra en el Banco, y el dinero es entregado al Cliente. En caso contrario finaliza el proceso habiéndose registrado tanto la solicitud del Cliente como la resolución del Banco.

**Sección 2 – Definición de servicios para la implementación del proceso de negocio descrito en la Sección 1 “Otorgar Préstamo” de un Banco. Diseño de servicios en SoaML.**

Esta sección trata sobre la definición de servicios para implementar el proceso de negocio “Otorgar Préstamos” según los diagramas que define el estándar SoaML. Cada test se compone de dos partes: una con diagramas SoaML que muestran los servicios definidos y demás elementos asociados, y otro con especificación textual de la obtención de servicios. Siga las instrucciones en cada parte para la realización de los tests, indicando en cada caso la hora inicio y la hora fin del mismo.

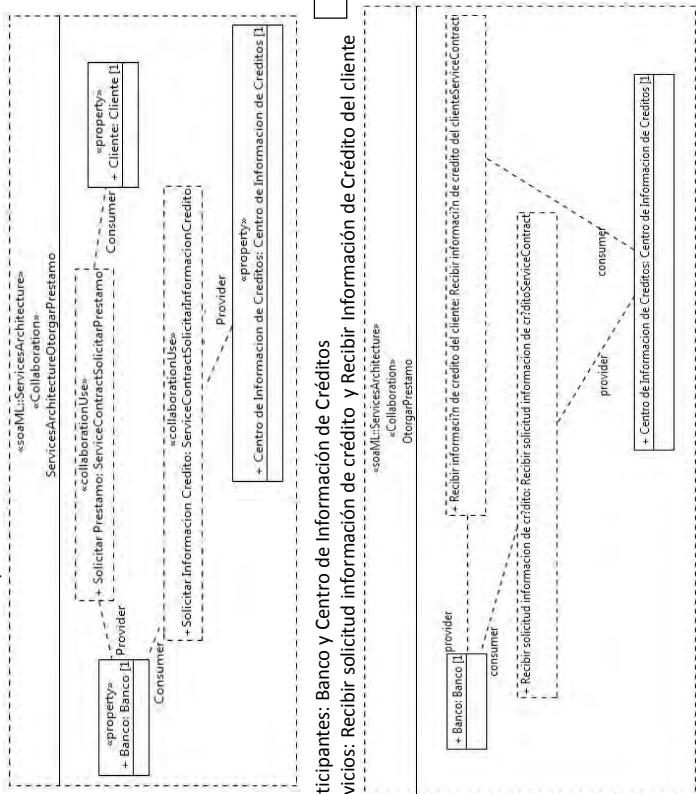
**1) Arquitectura de Servicios - Diagramas SoaML**

En esta sección se brindan diversos diagramas especificados en SoaML para especificar la Arquitectura de servicios que define los participantes, los servicios que proveen y consumen, y los roles definidos para la interacción de los participantes con los servicios, para implementar el proceso de negocio presentado en la Sección 1. Se quiere lograr el mayor grado de automatización posible.

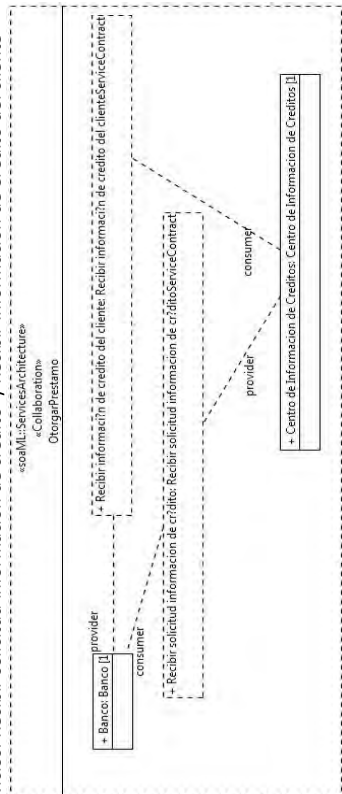
Ud. dispone de 4 opciones posibles para representar los elementos de SoaML que considera se corresponden con el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones.

hora inicio: hh:mm:ss \_\_\_\_\_

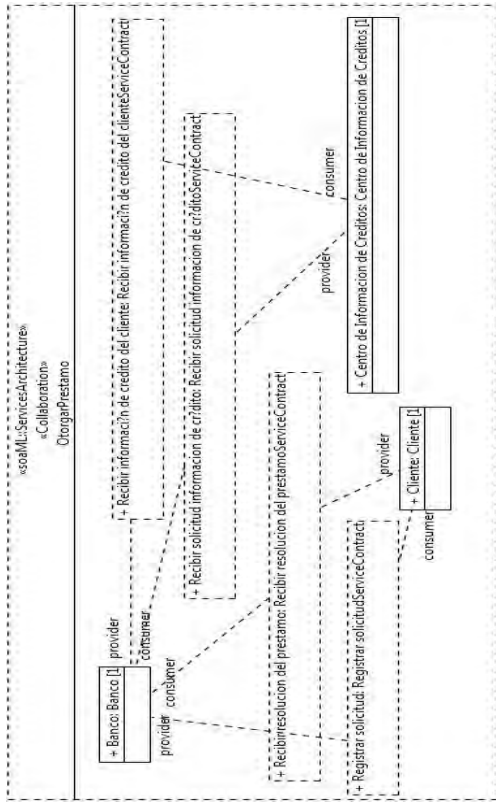
- a) Participantes: Banco, Cliente y Centro de Información de Créditos  
 Servicios: Solicitar Préstamo y Solicitar Información Crédito



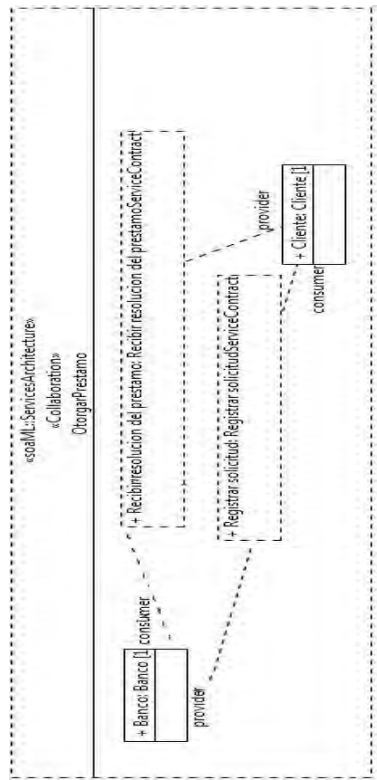
- b) Participantes: Banco y Centro de Información de Créditos  
 Servicios: Recibir solicitud información de crédito y Recibir Información de Crédito del cliente



- c) Participantes Banco, Cliente y Centro de Información de Créditos  
 Servicios: Registrar solicitud, Recibir resolución del préstamo, Recibir solicitud información de crédito y Recibir Información de Crédito del cliente



- d) Participantes: Banco y Cliente  
 Servicios: Registrar solicitud y Recibir resolución del préstamo



hora fin: hh:mm:ss \_\_\_\_\_

**1) Arquitectura de Servicios - Reglas de correspondencia**

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

**hora inicio:** hh:mm:ss \_\_\_\_\_  
 La Arquitectura de Servicios se corresponde con:

a) el proceso colaborativo "Otomar Préstamos" e incluye:  
 - los participantes: Cliente, Banco y Centro de Información de Crédito  
 - los servicios que se corresponden con las actividades involucradas en un flujo de mensaje entre dos procesos distintos:

- Recibir solicitud de información de crédito (proveedor Centro de Información de Crédito, consumidor Banco)
- Recibir información de crédito del cliente (proveedor Banco, consumidor Centro de Información de Crédito)
- Registrar solicitud (proveedor Banco, consumidor Cliente)
- Recibir resolución del préstamo (proveedor Cliente, consumidor Banco)

b) los procesos Banco y Cliente integrantes del proceso colaborativo "Otomar Préstamos" e incluye:  
 - los participantes: Banco y Cliente  
 - los servicios que se corresponden con las actividades involucradas en un flujo de mensaje entre los dos procesos:

- Registrar solicitud (proveedor Banco, consumidor Cliente)
- Recibir resolución del préstamo (proveedor Cliente, consumidor Banco)

c) los procesos Banco y Centro de Información de Crédito integrantes del proceso colaborativo "Otomar Préstamos" e incluye:  
 - los participantes: Banco y Centro de Información de Crédito

- los servicios: que se corresponden con las actividades involucradas en un flujo de mensaje entre los dos procesos:
- Recibir solicitud de información de crédito (proveedor Centro de Información de Crédito, consumidor Banco)
  - Recibir información de crédito del cliente (proveedor Banco, consumidor Centro de Información de Crédito)

d) el proceso colaborativo "Otomar Préstamos" e incluye:  
 - los participantes: Cliente, Banco y Centro de Información de Crédito  
 - los servicios que se corresponden con la combinación conceptual de actividades involucradas en un flujo de mensaje entre dos procesos distintos:  
 - Solicitar Préstamo (proveedor Banco, consumidor Cliente)  
 - Solicitar Información Crédito (proveedor Centro de Información de Crédito, consumidor Banco)

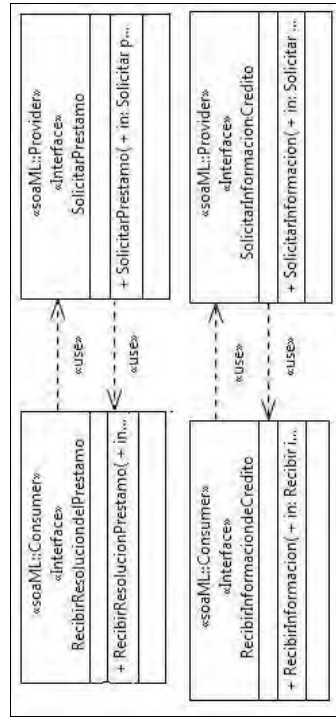
**hora fin:** hh:mm:ss \_\_\_\_\_

**2) Interfaces de Servicios - Diagramas SoaML**

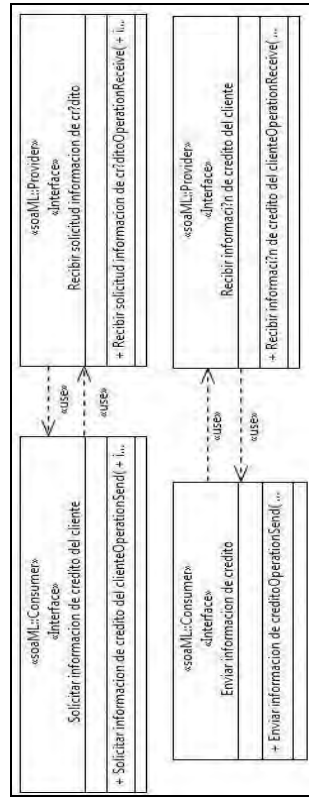
En esta sección se brindan diversos diagramas en SoaML para especificar las Interfaces con Operaciones y parámetros para los servicios definidos para implementar el proceso de negocio presentado en la Sección 1. El patrón de comunicación definido entre servicios corresponde al bidireccional con interfaces simple UML.

Ud. dispone de 4 opciones posibles para representar las Interfaces con SoaML para diseñar los servicios que se corresponden con el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones marcando la casilla correspondiente con una cruz.

**hora inicio:** hh:mm:ss \_\_\_\_\_  
 a) Interfaces para los Servicios: Solicitar Préstamo y Solicitar Información Crédito



b) Interfaces para los Servicios: Recibir solicitud Información de Crédito y Recibir información de Crédito del cliente



2) Interfaces de Servicios - Reglas de correspondencia

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

hora inicio: hh:mm:ss \_\_\_\_\_

Para cada servicio definido las interfaces se modelan como una interface para el proveedor y otra interface para el consumidor, con una operación cada una para ser invocada por la otra, teniendo en cuenta:

- a) las actividades involucradas en un flujo de mensaje entre los procesos Banco, Cliente y Centro de Información de Crédito del proceso colaborativo "Otomar Préstamos":
  - Recibir solicitud Información de Crédito (Interface provider Recibir solicitud información de Crédito, Interface consumer Solicitar información de crédito del cliente)
  - Recibir información de Crédito del cliente (Interface provider Recibir información de Crédito del cliente, Interface consumer Enviar información de crédito)
  - Registrar solicitud (Interface provider Registrar solicitud, interface consumer Solicitar Préstamo)
  - Recibir resolución del préstamo (Interface provider Recibir resolución del préstamo, interface consumer Enviar resolución del préstamo)

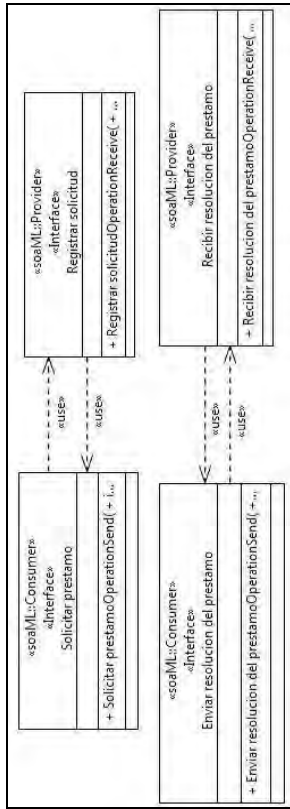
- b) las actividades involucradas en un flujo de mensajes entre los procesos Banco y Cliente del proceso colaborativo "Otomar Préstamos":
  - Registrar solicitud (Interface provider Registrar solicitud, Interface consumer Solicitar Préstamo)
  - Recibir resolución del préstamo (Interface provider Recibir resolución del préstamo, Interface consumer Enviar resolución del préstamo)

- c) las actividades involucradas en un flujo de mensajes entre los procesos Banco y Centro de Información de Crédito del proceso colaborativo "Otomar Préstamos":
  - Recibir solicitud Información de Crédito (Interface provider Recibir solicitud información de Crédito, Interface consumer Solicitar información de crédito del cliente)
  - Recibir información de Crédito del cliente (Interface provider Recibir información de Crédito del cliente, Interface consumer Enviar información de Crédito)

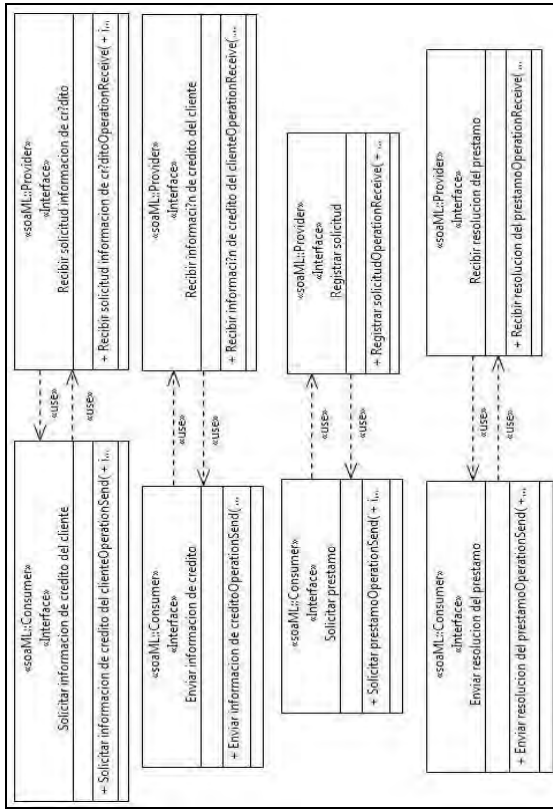
- d) las actividades combinadas involucradas en un flujo de mensajes entre los procesos Banco, Cliente y Centro de Información de Crédito del proceso colaborativo "Otomar Préstamos":
  - Solicitar Préstamo (Interface provider Solicitar Préstamo, interface consumer Recibir resolución del préstamo)
  - Solicitar información Crédito (Interface provider Solicitar información Crédito, Interface consumer Recibir información de crédito)

hora fin: hh:mm:ss \_\_\_\_\_

c) Interfaces para los Servicios: Registrar solicitud y Recibir resolución del préstamo



d) Interfaces para los Servicios: Recibir solicitud Información de Crédito, Recibir información de Crédito del cliente, Registrar solicitud, Recibir resolución del préstamo



hora fin: hh:mm:ss \_\_\_\_\_

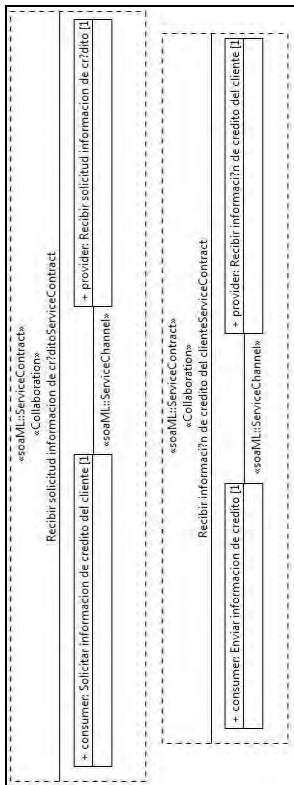
3) Contratos de Servicios - Diagramas SoaML

En esta sección se brindan diversos diagramas en SoaML para especificar las Interfaces con Operaciones y parámetros para los servicios definidos para implementar el proceso de negocio presentado en la Sección 1. El patrón de comunicación definido entre servicios corresponde al bidireccional con interfaces simple UML.

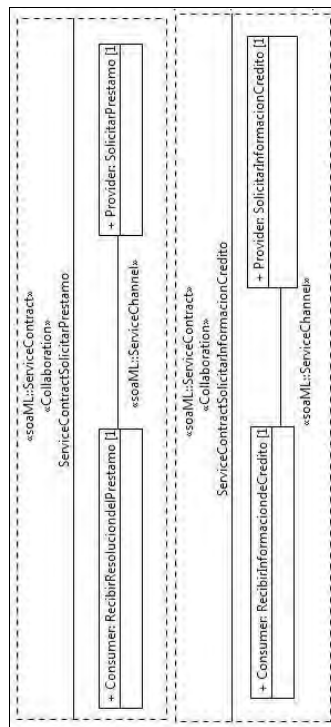
Ud. dispone de 4 opciones posibles para representar los contratos de servicios con SoaML para especificar los servicios que se corresponden con el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones marcando la casilla correspondiente con una cruz.

hora inicio: hh:mm:ss

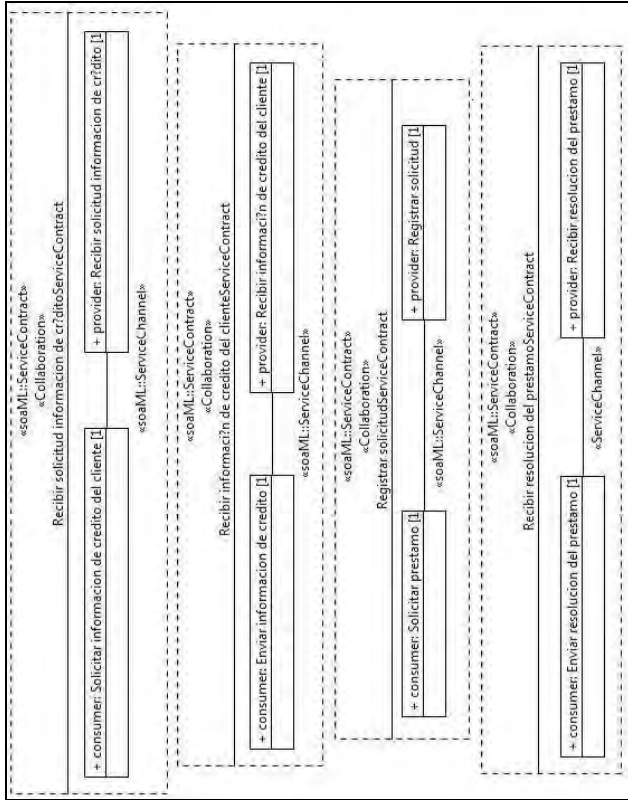
a) Contratos de Servicios para: Recibir solicitud Información de Crédito y Recibir información de Crédito del cliente



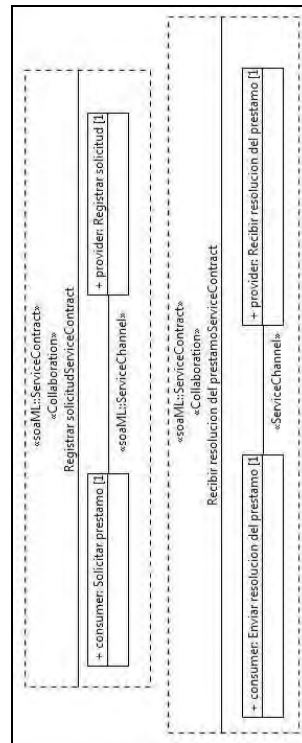
b) Contratos de Servicios para: Solicitar Préstamo y Solicitar Información Crédito



c) Contratos de Servicios: Recibir solicitud Información de Crédito, Recibir información de Crédito del cliente, Registrar solicitud, Registrar resolución del préstamo



d) Contratos de Servicios para: Registrar solicitud y Recibir resolución del préstamo



hora fin: hh:mm:ss

3) Contratos de Servicios - Reglas de correspondencia

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

hora inicio: hh:mm:ss \_\_\_\_\_  
 Para cada servicio definido con interfaces especificadas, el contrato de servicios indica los roles "consumer" y "provider" de tipo:

- a) Recibir solicitud Información de Crédito (rol provider de tipo Recibir solicitud información de Crédito, rol consumer de tipo Solicitar información de crédito del cliente)
  - Recibir información de Crédito del cliente (rol provider de tipo Recibir información de Crédito del cliente, rol consumer de tipo Enviar información de crédito)
  - Registrar solicitud (rol provider de tipo Registrar solicitud, rol consumer de tipo Solicitar Préstamo)
  - Recibir resolución del préstamo (rol provider de tipo Recibir resolución del préstamo, rol consumer de tipo Enviar resolución del préstamo)
  - b) Registrar solicitud (rol provider de tipo Registrar solicitud, rol consumer de tipo Solicitar Préstamo)
  - Recibir resolución del préstamo (rol provider de tipo Recibir resolución del préstamo, rol provider de tipo Enviar resolución del préstamo)
  - c) Recibir solicitud Información de Crédito (rol provider de tipo Recibir solicitud información de Crédito, rol consumer de tipo Solicitar información de crédito del cliente)
  - Recibir información de Crédito del cliente (rol provider de tipo Recibir información de Crédito del cliente, rol consumer de tipo Enviar información de Crédito)
  - d) Solicitar Préstamo (rol provider de tipo Solicitar Préstamo, rol consumer de tipo Recibir resolución del préstamo)
  - Solicitar información Crédito (rol provider de tipo Solicitar información Crédito, rol consumer de tipo Recibir información de crédito)
- hora fin: hh:mm:ss \_\_\_\_\_

4) Participantes y servicios asociados - Diagramas SoaML

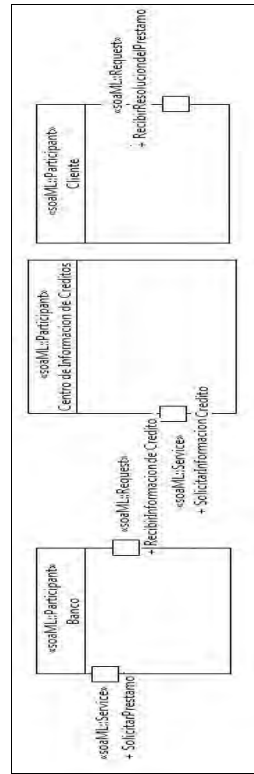
En esta sección se brindan diversos diagramas en SoaML para especificar los Participantes y sus puertos asociados a los servicios provistos y consumidos definidos para implementar el proceso de negocio presentado en la Sección 1. El patrón de comunicación definido entre servicios corresponde al bidireccional con interfaces simple UML.

Ud. dispone de 4 opciones posibles para representar los Participantes y sus puertos asociados con SoaML según la definición de servicios para el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones marcando la casilla correspondiente con una cruz.

hora inicio: hh:mm:ss \_\_\_\_\_

a) Participantes: Banco y Centro de Información de Crédito   
 Servicios: Recibir solicitud Información de Crédito y Recibir información de Crédito del cliente

b) Participantes: Banco, Cliente y Centro de Información de Crédito   
 Servicios: Solicitar Préstamo y Solicitar Información Crédito



4) Participantes y servicios asociados - Reglas de correspondencia

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

hora inicio: hh:mm:ss \_\_\_\_\_

Los participantes se corresponden con los procesos:

- a) Banco y Cliente del proceso colaborativo "Otorgar Préstamo", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:
  - Registrar solicitud (puerto Service Registrar solicitud, puerto Request Solicitar Préstamo)
  - Recibir resolución del préstamo (puerto Service Recibir resolución del préstamo, puerto Request Enviar resolución del préstamo)

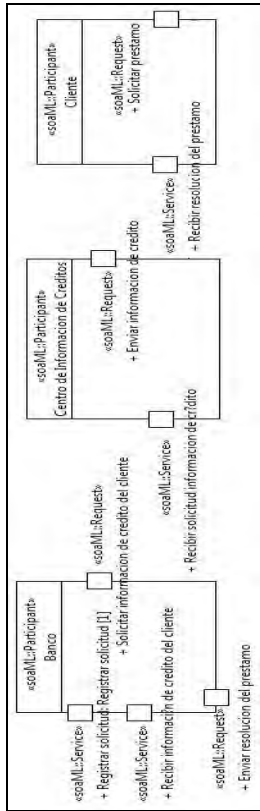
- b) Banco, Cliente y Centro de Información de Crédito del proceso colaborativo "Otorgar Préstamo", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:
  - Recibir solicitud información de Crédito (puerto Service Recibir solicitud información de Crédito, puerto Request Solicitar información de crédito del cliente)
  - Recibir información de Crédito del cliente (puerto Service Recibir información de Crédito del cliente, puerto Request Enviar información de crédito)
  - Registrar solicitud (puerto Service Registrar solicitud, puerto Request Solicitar Préstamo)
  - Recibir resolución del préstamo (puerto Service Recibir resolución del préstamo, puerto Request Enviar resolución del préstamo)

- c) Banco y Centro de Información de Crédito del proceso colaborativo "Otorgar Préstamo", los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:
  - Recibir solicitud información de Crédito (puerto Service Recibir solicitud información de Crédito, puerto Consumer Solicitar información de crédito del cliente)
  - Recibir información de Crédito del cliente (puerto Service Recibir información de Crédito del cliente, puerto Request Enviar información de Crédito)

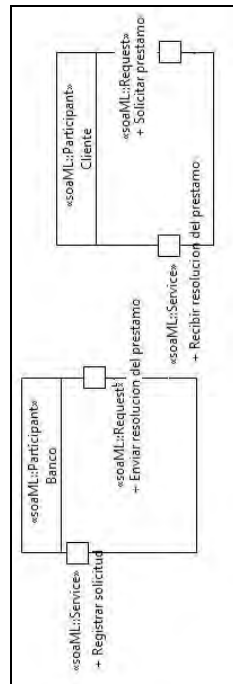
- d) Banco, Cliente y Centro de Información de Crédito del proceso colaborativo "Otorgar Préstamos", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:
  - Solicitar Préstamo (puerto Service Solicitar Préstamo, puerto Request Recibir resolución del préstamo)
  - Solicitar información Crédito (puerto Service Solicitar información Crédito, puerto Request Recibir información de crédito)

hora fin: hh:mm:ss \_\_\_\_\_

- c) Participantes: Banco, Cliente y Centro de Información de Crédito  
 Servicios: Recibir solicitud información de Crédito, Recibir información de Crédito del cliente, Registrar solicitud, Recibir resolución del préstamo



- d) Participantes: Banco y Cliente  
 Servicios: Registrar solicitud y Recibir resolución del préstamo



hora fin: hh:mm:ss \_\_\_\_\_



**Descripción del proceso de negocio "Licencia de conducir"**

El Solicitante solicita una cita para Licencia de conducir en el formulario de "Solicitud de cita para Licencia de conducir" del sitio web de la Oficina de Tráfico, para lo cual ingresa sus datos y envía la solicitud.

La Oficina de Tráfico registra la solicitud ingresada y asigna y envía al Solicitante la fecha de cita para la Licencia de conducir. En el día y hora asignados el Solicitante se presenta en la Oficina de Tráfico con la documentación requerida, la cual es verificada por la Oficina para luego cargar el monto a pagar por el trámite y la licencia de conducir a emitir. El Solicitante puede pagar en efectivo o a crédito, en el segundo caso la Oficina de Tráfico se comunica con la Entidad de crédito para realizar el cargo correspondiente. Si hubiera algún problema con el crédito el usuario podrá cambiar de forma de pago intentando con otra tarjeta o finalmente con la opción de efectivo si ninguna es aprobada.

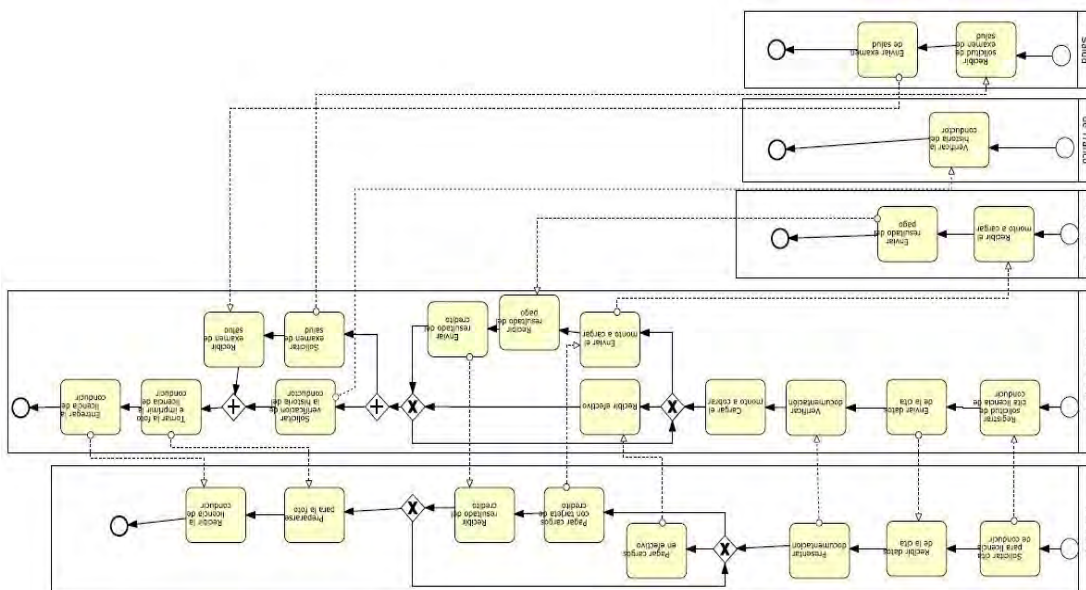
Luego que el pago es realizado (en efectivo o crédito aprobado), la Oficina de Tráfico realiza en paralelo dos comprobaciones sobre el Solicitante:

- 1 - solicitar información de la historia de conducción del Solicitante, si la tuviera, a la Oficina Central de Tráfico (multas pendientes, puntos perdidos, etc)
- 2 - solicitar el documento del examen de salud del organismo de salud donde fue realizado según indicación del usuario

Si ambos resultados son correctos se toma la foto del Solicitante y se imprime la licencia de conducir, que le es entregada. El caso no exitoso no está modelado en el diagrama del Proceso de negocio, pero requerirá nuevas acciones antes de poder otorgar la licencia de conducir.

**MATERIAL EXPERIMENTAL – GRUPO A**

**Sección 1 – Planteamiento del problema: descripción del Proceso de negocio colaborativo "Licencia de conducir" de una Oficina de Tráfico.**



**Sección 2 – Definición de servicios para la implementación del proceso de negocio descrito en la Sección 1. “Licencia de conducir” de una Oficina de Tránsito. Diseño de servicios en SoaML.**

Esta sección trata sobre la definición de servicios para implementar el proceso de negocio “Licencia de conducir” según los diagramas que define el estándar SoaML. Cada test se compone de dos partes: una con especificación textual de la obtención de servicios, y otra con diagramas SoaML que muestran los servicios definidos y demás elementos asociados. Siga las instrucciones en cada parte para la realización de los tests, indicando en cada caso la hora inicio y la hora fin del mismo.

**1) Arquitectura de Servicios - Reglas de correspondencia**

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

**hora inicio:** hh:mm:ss \_\_\_\_\_  
 La Arquitectura de Servicios se corresponde con:

- a) el proceso colaborativo “Licencia de conducir” e incluye:
- los participantes: Solicitante, Oficina de Tránsito, Entidad de crédito, Oficina Central de Tránsito, y Compañía de salud.
  - los servicios que se corresponden con las actividades involucradas en un flujo de mensaje entre dos procesos distintos:
    - Registrar solicitud de licencia de conducir (proveedor Oficina de Tránsito, consumidor Solicitante)
    - Recibir datos de la cita (proveedor Solicitante, consumidor Oficina de Tránsito)
    - Recibir el monto a cargar (proveedor Entidad de crédito, consumidor Oficina de Tránsito)
    - Recibir resultado del pago (proveedor Oficina de Tránsito, consumidor Entidad de crédito)
    - Recibir solicitud de examen de salud (proveedor Compañía de salud, consumidor Oficina de Tránsito)
    - Recibir examen de salud (proveedor Oficina de Tránsito, consumidor Compañía de salud)
    - Verificar la historia del conductor (proveedor Oficina Central de Tránsito, consumidor Oficina de Tránsito)

- b) los procesos Solicitante, Oficina de Tránsito y Entidad de crédito integrantes del proceso colaborativo “Licencia de conducir” e incluye:
- los participantes: Solicitante, Oficina de Tránsito y Entidad de crédito
  - los servicios que se corresponden con las actividades involucradas en un flujo de mensaje entre los dos procesos:
    - Registrar solicitud de licencia de conducir (proveedor Oficina de Tránsito, consumidor Solicitante)
    - Recibir datos de la cita (proveedor Solicitante, consumidor Oficina de Tránsito)
    - Recibir el monto a cargar (proveedor Entidad de crédito, consumidor Oficina de Tránsito)
    - Recibir resultado del pago (proveedor Oficina de Tránsito, consumidor Entidad de crédito)

- 
- c) los procesos Oficina de Tránsito, Oficina Central de Tránsito, Compañía de salud y Entidad de Crédito integrantes del proceso colaborativo “Licencia de conducir” e incluye:
    - los participantes: Oficina de Tránsito, Oficina Central de Tránsito, Compañía de salud y Entidad de Crédito
    - los servicios: que se corresponden con actividades y combinación conceptual de actividades involucradas en un flujo de mensaje entre los dos procesos:
      - Solicitud de monto a cargar (proveedor Entidad de crédito, consumidor Oficina de Tránsito)
      - Verificar examen de salud (proveedor Compañía de salud, consumidor Oficina de Tránsito)
      - Verificar la historia del conductor (proveedor Oficina central de Tránsito, consumidor Oficina de Tránsito)

- 
- d) el proceso colaborativo “Licencia de conducir” e incluye:
    - los participantes: Solicitante, Oficina de Tránsito, Entidad de crédito, Oficina Central de Tránsito, y Compañía de salud.
    - los servicios: que se corresponden con actividades y combinación conceptual de actividades involucradas en un flujo de mensaje entre los dos procesos
      - Solicitud de licencia de conducir (proveedor Oficina de Tránsito, consumidor Solicitante)
      - Solicitud de monto a cargar (proveedor Entidad de crédito, consumidor Oficina de Tránsito)
      - Verificar examen de salud (proveedor Compañía de salud, consumidor Oficina de Tránsito)
      - Verificar la historia del conductor (proveedor Oficina Central de Tránsito, consumidor Oficina de Tránsito)

**hora fin:** hh:mm:ss \_\_\_\_\_

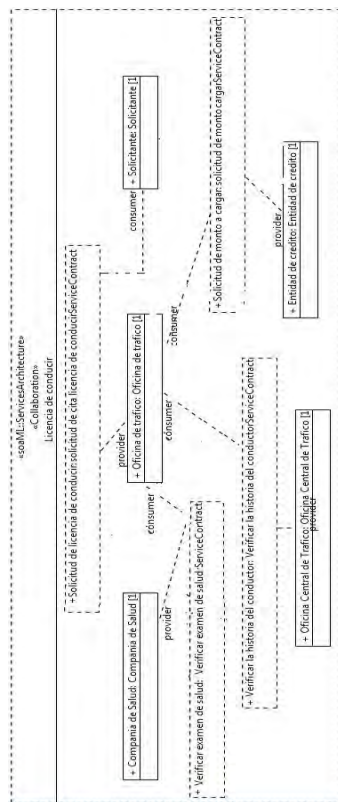
1) **Arquitectura de Servicios - Diagramas SoaML**

En esta sección se brindan diversos diagramas especificados en SoaML para especificar la Arquitectura de servicios que define los participantes, los servicios que proveen y consumen, y los roles definidos para la interacción de los participantes con los servicios, para implementar el proceso de negocio presentado en la Sección 1. Se quiere lograr el mayor grado de automatización posible.

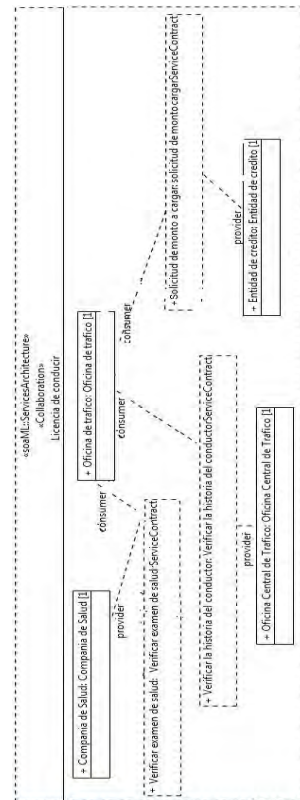
Ud. dispone de 4 opciones posibles para representar los elementos de SoaML que considera se corresponden con el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones.

hora inicio: hh:mm:ss \_\_\_\_\_

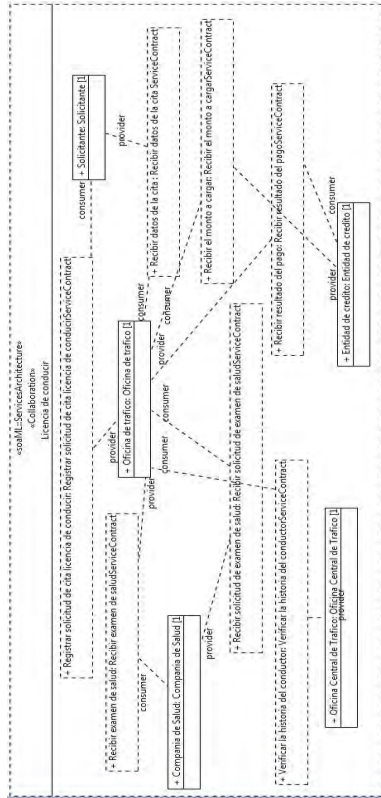
- a) Participantes: Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina central de Tráfico, Compañía de salud  
 Servicios: Solicitud de Licencia de conducir, Solicitud de monto a cargar, Verificar examen de salud, Verificar la historia del conductor.



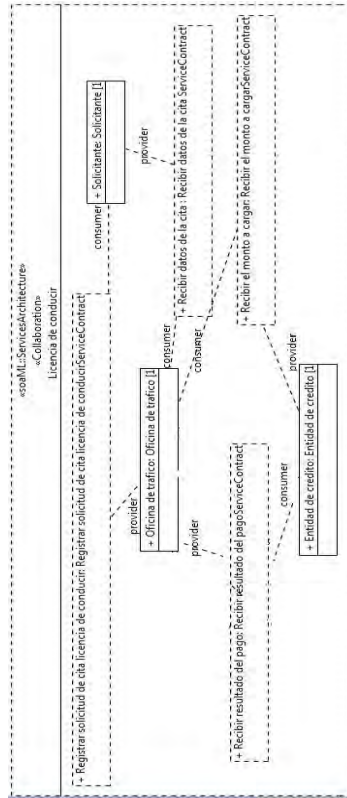
- b) Participantes: Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico, Compañía de salud  
 Servicios: Solicitud de monto a cargar, Verificar examen de salud, Verificar la historia del conductor.



- c) Participantes: Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina central de Tráfico, Compañía de salud  
 Servicios: Registrar solicitud de Licencia de conducir, Recibir datos de la cita, Recibir el monto a cargar, Recibir resultado del pago, Recibir solicitud de examen de salud, Recibir examen de salud, Verificar la historia del conductor.



- d) Participantes: Solicitante, Oficina de Tráfico, Entidad de crédito,  
 Servicios: Registrar solicitud de cita licencia de conducir, Recibir datos de la cita, Recibir el monto a cargar y Recibir resultado del pago.



hora fin: hh:mm:ss \_\_\_\_\_

**2) Interfaces de Servicios - Reglas de correspondencia**

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

**hora inicio:** hh:mm:ss \_\_\_\_\_

Para cada servicio definido las interfaces se modelan como una interface para el proveedor y otra interface para el consumidor, con una operación cada una para ser invocada por la otra, teniendo en cuenta:

- a) las actividades involucradas en un flujo de mensaje entre los procesos Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico, y Compañía de salud del proceso colaborativo "Licencia de conducir":
- Solicitud de licencia de conducir (interface provider Solicitar licencia de conducir, interface consumer Solicitar licencia de conducir)
  - Solicitud de monto a cargar (interface provider Verificar examen de salud, interface consumer Solicitar examen de salud)
  - Verificar examen de salud (interface provider Solicitar monto a cargar, interface consumer Enviar el monto a cargar)
  - Verificar la historia del conductor (interface provider Verificar la historia del conductor, interface consumer Solicitar verificación de la historia del conductor)

- b) las actividades involucradas en un flujo de mensajes entre los procesos Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico, y Compañía de salud del proceso colaborativo "Licencia de conducir":
- Registrar solicitud de cita licencia de conducir (interface provider Registrar solicitud de cita licencia de conducir, interface consumer Solicitar cita para licencia de conducir)
  - Recibir datos de la cita (interface provider Recibir datos de la cita, interface consumer Enviar datos de la cita)
  - Recibir el monto a cargar (interface provider Recibir el monto a cargar, interface consumer Enviar el monto a cargar)
  - Recibir resultado del pago (interface provider Recibir resultado del pago, interface consumer Enviar resultado del pago)
  - Recibir solicitud de examen de salud (interface provider Recibir solicitud de examen de salud, interface consumer Solicitar examen de salud)
  - Recibir examen de salud (interface provider Recibir examen de salud, interface consumer Enviar examen de salud)
  - Verificar la historia del conductor (interface provider Verificar la historia del conductor, interface consumer Solicitar verificación de la historia del conductor)

- c) las actividades involucradas en un flujo de mensajes entre los procesos Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico, y Compañía de salud del proceso colaborativo "Licencia de conducir":
- Solicitud de monto a cargar (interface provider Solicitar monto a cargar, interface consumer Enviar el monto a cargar)
  - Verificar examen de salud (interface provider Verificar examen de salud, interface consumer Solicitar examen de salud)
  - Verificar la historia del conductor (interface provider Verificar la historia del conductor, interface consumer Solicitar verificación de la historia del conductor)

- d) las actividades combinadas involucradas en un flujo de mensajes entre los procesos Solicitante, Oficina de Tráfico, Entidad de crédito del proceso colaborativo "Licencia de conducir":
- Registrar solicitud de cita de licencia de conducir (interface provider Registrar solicitud de cita de licencia de conducir, interface consumer Solicitar cita para licencia de conducir)
  - Recibir datos de la cita (interface provider Recibir datos de la cita, interface consumer Enviar datos de la cita)
  - Recibir el monto a cargar (interface provider Recibir el monto a cargar, interface consumer Enviar el monto a cargar)
  - Recibir resultado del pago (interface provider Recibir resultado del pago, interface consumer Enviar resultado del pago)

**hora fin:** hh:mm:ss \_\_\_\_\_

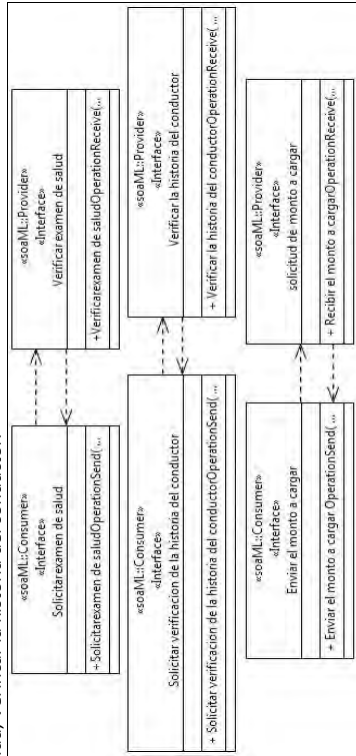
2) Interfaces de Servicios - Diagramas SoaML

En esta sección se brindan diversos diagramas en SoaML para especificar las Interfaces con Operaciones y parámetros para los servicios definidos para implementar el proceso de negocio presentado en la Sección 1. El patrón de comunicación definido entre servicios corresponde al bidireccional con interfaces simple UML.

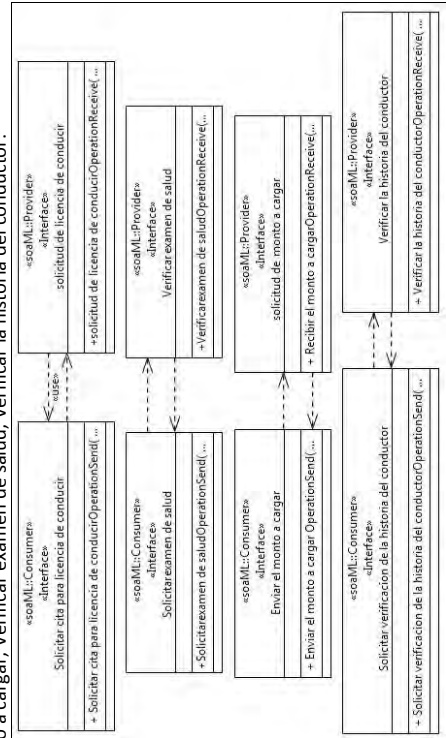
Ud. dispone de 4 opciones posibles para representar las Interfaces con SoaML para diseñar los servicios que se corresponden con el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones marcando la casilla correspondiente con una cruz.

hora inicio: hh:mm:ss

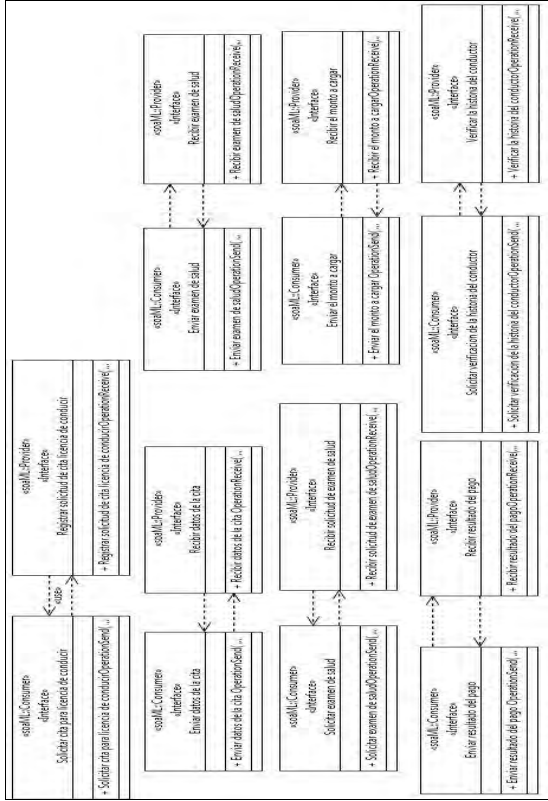
a) Interfaces para los Servicios: Solicitud de monto a cargar, Verificar examen de salud, Verificar la historia del conductor.



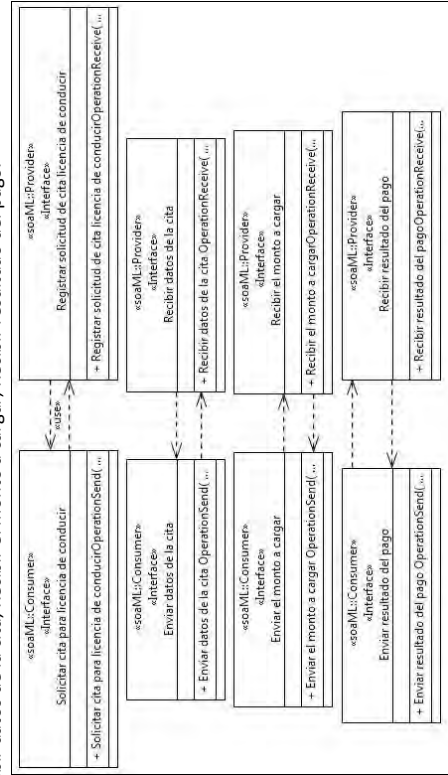
b) Interfaces para los Servicios: Solicitud de licencia de conducir, Solicitud de monto a cargar, Verificar examen de salud, Verificar la historia del conductor.



c) Interfaces para los Servicios: Registrar solicitud de cita de licencia de conducir, Recibir datos de la cita, Recibir el monto a cargar, Recibir resultado del pago, Recibir solicitud de examen de salud, Recibir examen de salud, Verificar la historia del conductor.



d) Interfaces para los Servicios: Registrar solicitud de cita de licencia de conducir, Recibir datos de la cita, Recibir el monto a cargar, Recibir resultado del pago.



hora fin: hh:mm:ss

**3) Contratos de Servicios - Reglas de correspondencia**

En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

hora inicio: hh:mm:ss \_\_\_\_\_

Para cada servicio definido con interfaces especificadas, el contrato de servicios indica los roles "consumer" y "provider" de tipo:

a) - Registrar solicitud de cita licencia de conducir (rol provider de tipo )  
 Registrar solicitud de cita licencia de conducir, rol consumer de tipo Solicitar cita para licencia de conducir  
 - Recibir datos de la cita (rol provider de tipo Recibir datos de la cita, rol consumer de tipo Enviar datos de la cita)  
 - Recibir el monto a cargar (rol provider de tipo Recibir el monto a cargar, rol consumer de tipo Enviar el monto a cargar)  
 - Recibir resultado del pago (rol provider de tipo Recibir resultado del pago, rol consumer de tipo Enviar resultado del pago)  
 - Recibir solicitud de examen de salud (rol provider de tipo Recibir solicitud de examen de salud, rol consumer de tipo Solicitar examen de salud)  
 - Recibir examen de salud (rol provider de tipo Recibir examen de salud, rol consumer de tipo Enviar examen de salud)  
 - Verificar la historia del conductor (rol provider de tipo Verificar la historia del conductor, rol consumer de tipo Solicitar verificación de la historia del conductor)

b) - Solicitud de licencia de conducir (rol provider de tipo Solicitud de licencia de conducir, rol consumer de tipo Solicitar licencia de conducir)  
 - Solicitud de monto a cargar (rol provider de tipo Verificar examen de salud, rol consumer de tipo Solicitar examen de salud)  
 - Verificar examen de salud (rol provider de tipo Solicitud de monto a cargar, rol consumer de tipo Enviar el monto a cargar)  
 - Verificar la historia del conductor (rol provider de tipo Verificar la historia del conductor, rol consumer de tipo Solicitar verificación de la historia del conductor)

c) - Registrar solicitud de cita de licencia de conducir (rol provider de tipo )  
 Registrar solicitud de cita de licencia de conducir, rol consumer de tipo Solicitar cita para licencia de conducir)  
 - Recibir datos de la cita (rol provider de tipo Recibir datos de la cita, rol consumer de tipo Enviar datos de la cita)  
 - Recibir el monto a cargar (rol provider de tipo Recibir el monto a cargar, rol consumer de tipo Enviar el monto a cargar)  
 - Recibir resultado del pago (rol provider de tipo Recibir resultado del pago, rol consumer de tipo Enviar resultado del pago)

d) - Solicitud de monto a cargar (rol provider de tipo Solicitud de monto a cargar, rol consumer de tipo Enviar el monto a cargar)  
 - Verificar examen de salud (rol provider de tipo Verificar examen de salud, rol consumer de tipo Solicitar examen de salud)  
 - Verificar la historia del conductor (rol provider de tipo Verificar la historia del conductor, rol consumer de tipo Solicitar verificación de la historia del conductor)

hora fin: hh:mm:ss \_\_\_\_\_

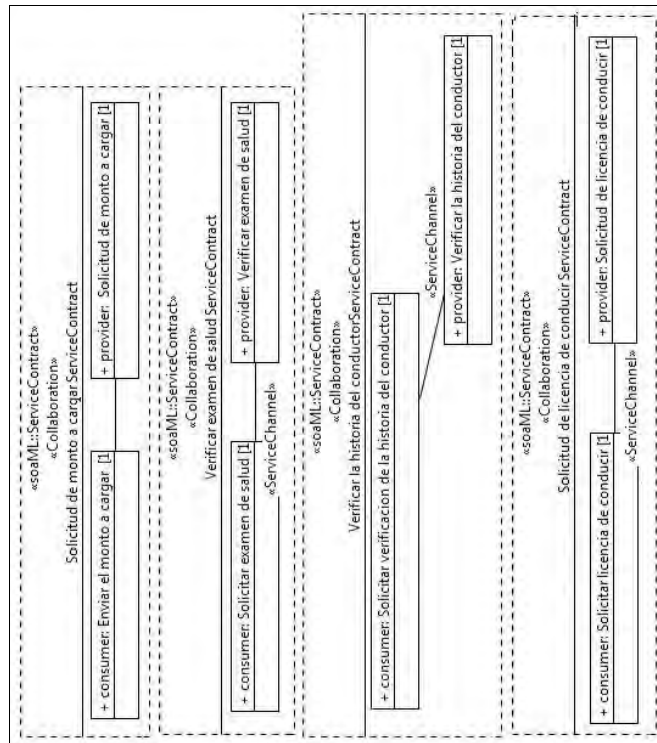
3) **Contratos de Servicios - Diagramas SoaML**

En esta sección se brindan diversos diagramas en SoaML para especificar las Interfaces con Operaciones y parámetros para los servicios definidos para implementar el proceso de negocio presentado en la Sección 1. El patrón de comunicación definido entre servicios corresponde al bidireccional con interfaces simple UML.

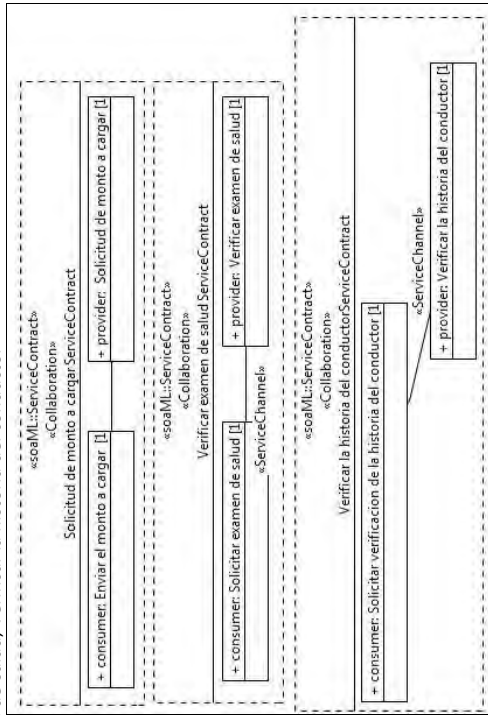
Ud. dispone de 4 opciones posibles para representar los contratos de servicios con SoaML para especificar los servicios que se corresponden con el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones marcando la casilla correspondiente con una cruz.

hora inicio: hh:mm:ss \_\_\_\_\_

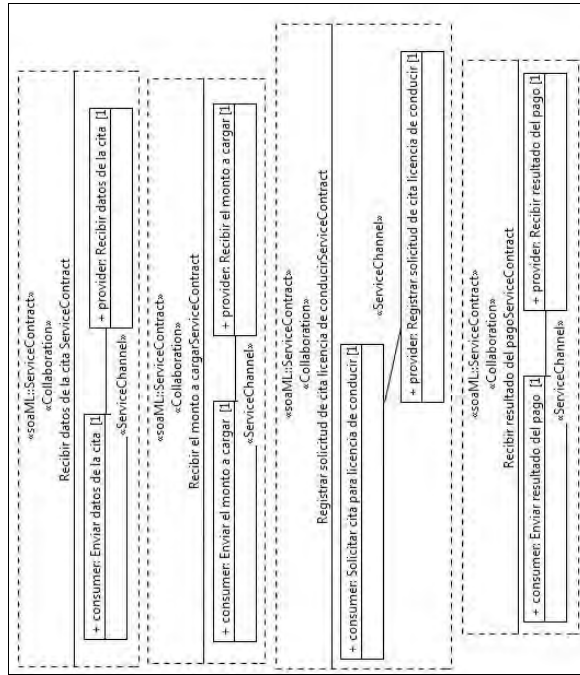
a) **Contratos de Servicios para: Solicitud de monto a cargar, Verificar examen de salud, Verificar la historia del conductor, Solicitud de cita licencia de conducir**



b) **Contratos de Servicios para: Solicitud de monto a cargar, Verificar examen de salud, Verificar la historia del conductor**



c) **Contratos de Servicios: Recibir datos de la cita, Recibir el monto a cargar, Registrar solicitud de cita licencia de conducir, Recibir resultado del pago**



Appendix D. Experimental material of the QVT transformations validation experiment

4) Participantes y servicios asociados - Reglas de correspondencia

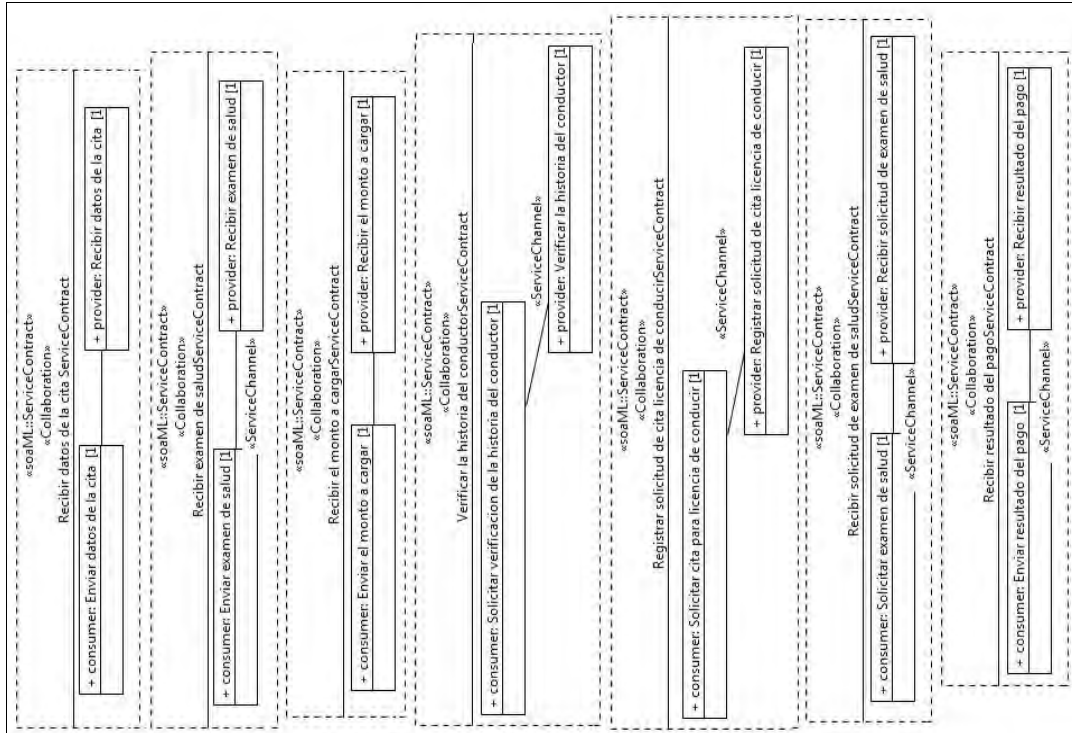
En esta sección se brindan diversas reglas de correspondencia que permiten obtener las distintas opciones para el modelado de servicios en presentadas en la parte 1). Ud. dispone de 4 opciones posibles que identifican correspondencias posibles entre los elementos del modelo del proceso de negocio y los elementos del modelo de servicios. Debe seleccionar la regla que identifica su preferencia.

hora inicio: hh:mm:ss \_\_\_\_\_

Los participantes se corresponden con los procesos:

- a) Solicitante, Entidad de crédito y Oficina de tráfico del proceso colaborativo "Licencia de conducir", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:
  - Recibir el monto a cargar (puerto Service Recibir el monto a cargar, puerto Request Enviar el monto a cargar)
  - Recibir datos de la cita (puerto Service Recibir datos de la cita, puerto Request Enviar datos de la cita)
  - Recibir resultado de pago (puerto Service Recibir resultado de pago, puerto Request Enviar resultado del pago)
  - Registrar solicitud de cita licencia de conducir (puerto Service Registrar solicitud de cita licencia de conducir, puerto Request Solicitar cita para licencia de conducir)
- b) Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico y Campaña de salud del proceso colaborativo "Licencia de conducir", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:
  - Recibir solicitud de examen de salud (puerto Service Recibir solicitud de examen de salud, puerto Request Solicitar examen de salud)
  - Recibir el monto a cargar (puerto Service Recibir el monto a cargar, puerto Request Enviar el monto a cargar)
  - Recibir datos de la cita (puerto Service Recibir datos de la cita, puerto Request Enviar datos de la cita)
  - Verificar la historia del conductor (puerto Service Verificar la historia del conductor, puerto Request Solicitar verificación de la historia del conductor)
  - Registrar solicitud de cita licencia de conducir (puerto Service Registrar solicitud de cita licencia de conducir, puerto Request Solicitar cita para licencia de conducir)
  - Recibir examen de salud (puerto Service Recibir examen de salud, puerto Request Enviar examen de salud)
  - Recibir resultado del pago (puerto Recibir resultado del pago, puerto Request Enviar resultado de pago)

- d) Contratos de Servicios para: Recibir datos de la cita, Recibir examen de salud, Recibir el monto a cargar, Verificar la historia del conductor, Registrar solicitud de cita de licencia de conducir, Recibir solicitud de examen de salud, Recibir resultado del pago



hora fin: hh:mm:ss \_\_\_\_\_



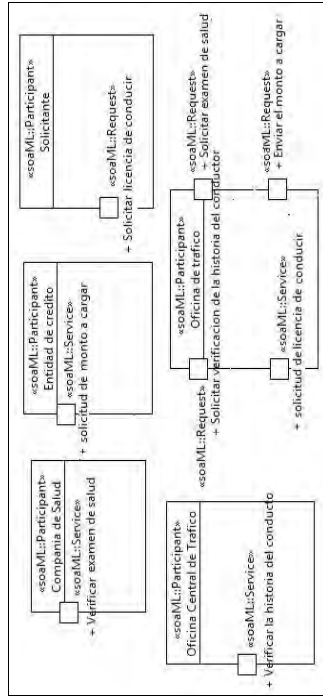
1) Participantes y servicios asociados - Diagramas SoaML

En esta sección se brindan diversos diagramas en SoaML para especificar los Participantes y sus puertos asociados a los servicios provistos y consumidos definidos para implementar el proceso de negocio presentado en la Sección 1. El patrón de comunicación definido entre servicios corresponde al bidireccional con interfaces simple UML.

Ud. dispone de 4 opciones posibles para representar los Participantes y sus puertos asociados con SoaML según la definición de servicios para el proceso de negocio descrito. Para indicar que opción prefiere debe elegir una sola de las opciones marcando la casilla correspondiente con una cruz.

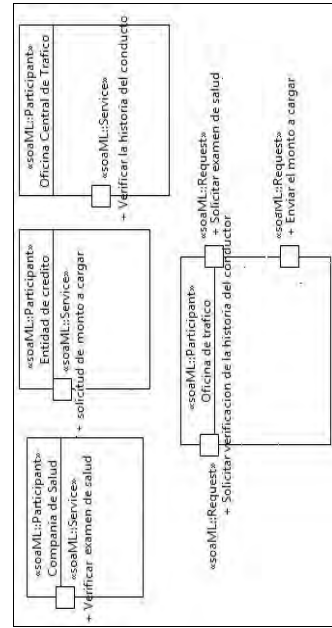
hora inicio: hh:mm:ss

- a) Participantes: Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico y Compañía de salud
- Servicios: Verificar examen de salud, Verificar la historia del conductor, Solicitar licencia de conducir, solicitud de monto a cargar



b) Participantes: Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico y Compañía de salud

- Servicios: Verificar examen de salud, Solicitar licencia de conducir, Verificar la historia del conductor



c) Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico y Compañía de salud del proceso colaborativo "Licencia de conducir", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:

- Verificar examen de salud (puerto Service Verificar examen de salud, puerto Request Solicitar examen de salud)
- Solicitud de monto a cargar (puerto Service Solicitud de monto a cargar, puerto Request Enviar el monto a cargar)
- Verificar la historia del conductor (puerto Verificar la historia del conductor, puerto Request Solicitar verificación de la historia del conductor)

d) Solicitante, Oficina de Tráfico, Entidad de crédito, Oficina Central de Tráfico y Compañía de salud del proceso colaborativo "Licencia de conducir", y los servicios que provee y consume cada uno se definen según los roles en el contrato de servicio y las interfaces especificadas:

- Verificar examen de salud (puerto Service Verificar examen de salud, puerto Request Solicitar examen de salud)
- Solicitud de monto a cargar (puerto Service Solicitud de monto a cargar, puerto Request Enviar el monto a cargar)
- Verificar la historia del conductor (puerto Verificar la historia del conductor, puerto Request Solicitar verificación de la historia del conductor)
- Solicitud de licencia de conducir (puerto Service Solicitud de licencia de conducir, puerto Request Solicitar licencia de conducir)

hora fin: hh:mm:ss \_\_\_\_\_



### **D.2.3. Part 2**

Part 2 of the experiment which evaluates the Understandability of the SoaML diagrams resulting from the QVT transformations, consisted in performing four exercises for each of the three BP models provided. Each exercise presented a SoaML diagram for which six questions had to be answered with True or False, about elements in the diagram.

In this part, for each BP model only the generated diagrams from the QVT transformations were provided, as we were evaluating the service models generated. The order in which the BP models were presented was different for each defined group A and B, as described before, and the questions for each exercise were randomized for each subject. After finishing each exercise the evaluation of the SoaML diagram complexity was also asked, in the defined scale from 1 to 5.



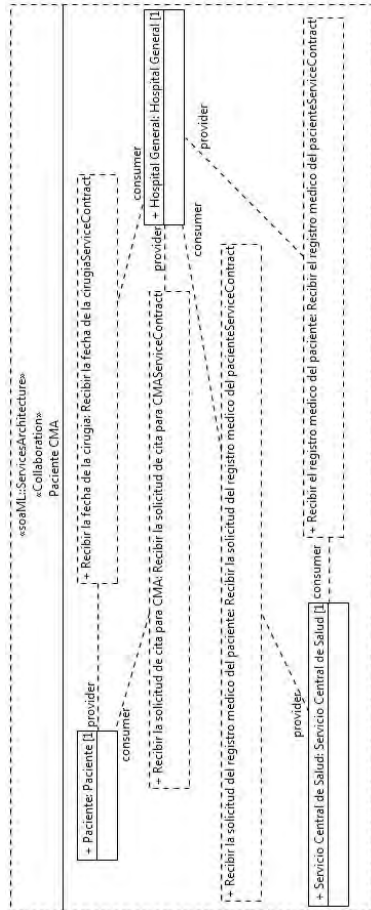
**Sección 2 – Definición de servicios para la implementación del proceso de negocio descrito en la Sección 1. “Paciente CMA” de un Hospital. Diseño de servicios en SoaML.**

En esta sección se brindan diversos diagramas especificados en SoaML para el modelado de los servicios que considera necesarios para implementar el proceso de negocio presentado en la Sección 1.

En cada pregunta se plantean 6 opciones relacionadas con el modelo de servicios que se presenta que ud. deberá responder con SI/NO, y una pregunta sobre la COMPLEJIDAD del modelo presentado, en la escala de 1 a 5 siendo: 1 – muy simple, 2 – algo simple, 3 – normal, 4 – algo complejo, 5 – muy complejo

**1. Arquitectura de Servicios**

Esta pregunta requiere identificar elementos relacionados con la Arquitectura de servicios definida para realizar el proceso de negocio “Paciente CMA”.



1.1) El participante Hospital General provee los servicios “Recibir la solicitud de cita para CMA”, “Recibir la solicitud del registro medico del paciente”, “Recibir la fecha de la cirugía”, “Recibir el registro médico del paciente”. V F

1.2) El rol provider en el contrato de servicio del servicio “Recibir la solicitud del registro médico del paciente” es jugado por el participante Servicio Central de Salud. V F

1.3) El participante Paciente juega el rol consumer definido en el contrato de servicio del servicio “Recibir la fecha de la cirugía”. V F

1.4) El contrato de servicio para el servicio “Recibir la solicitud de cita para CMA” define la interacción de los participantes Paciente y Hospital General con el servicio con los roles consumer y provider respectivamente. V F

1.5) Los roles definidos para el servicio “Recibir el registro médico del paciente” consumer y provider son jugados por los participantes Servicio Central de Salud y Hospital General respectivamente. V F

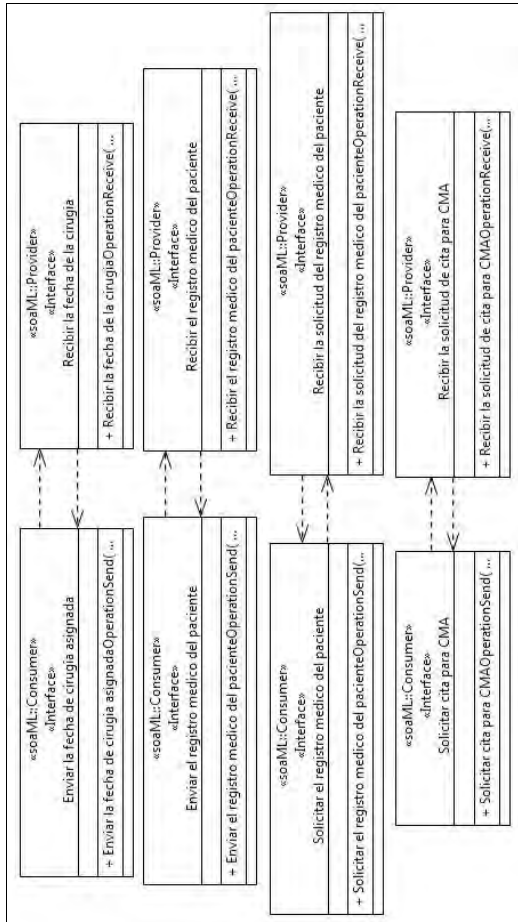
1.6) El participante Servicio Central de Salud interactúa con el participante Paciente en el servicio “Recibir la fecha de la cirugía” con el rol provider. V F

1.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo

**2) Interfaces de servicios**

Esta pregunta requiere identificar elementos relacionados con las interfaces provistas y requeridas definidas para los servicios del proceso de negocio “Paciente CMA”.



2.1) La interface “Recibir la fecha de la cirugía” provee la operación “Recibir la fecha de la cirugíaOperationReceive” que es invocada por la interface “Enviar la fecha de cirugía asignada”. V F

2.2) La interface “Solicitar cita para CMA” provee la operación “Solicitar cita para CMAOperation Send” que es invocada por el proveedor para iniciar la interacción con el servicio “Recibir la solicitud de cita para CMA”. V F

2.3) “Solicitar el registro medico del paciente” es la interface provista para ser invocada cuando se quiere obtener el registro médico de un paciente. V F

2.4) El servicio “Recibir la solicitud de cita para CMA” se define con las interfaces “Recibir la solicitud de cita para CMA” y “Solicitar cita para CMA”, que se invocan entre sí mediante las operaciones definidas en cada interface. V F

2.5) La interface “Recibir la solicitud del registro médico del paciente” invoca la operación “Solicitar el registro medico del pacienteOperation Send” para contestar en la interacción definida para el servicio. V F

2.6) La interface “Enviar el registro medico del paciente” es invocada por el Servicio Central de Salud para enviar la información del registro medico del paciente solicitada por el Hospital General. V F

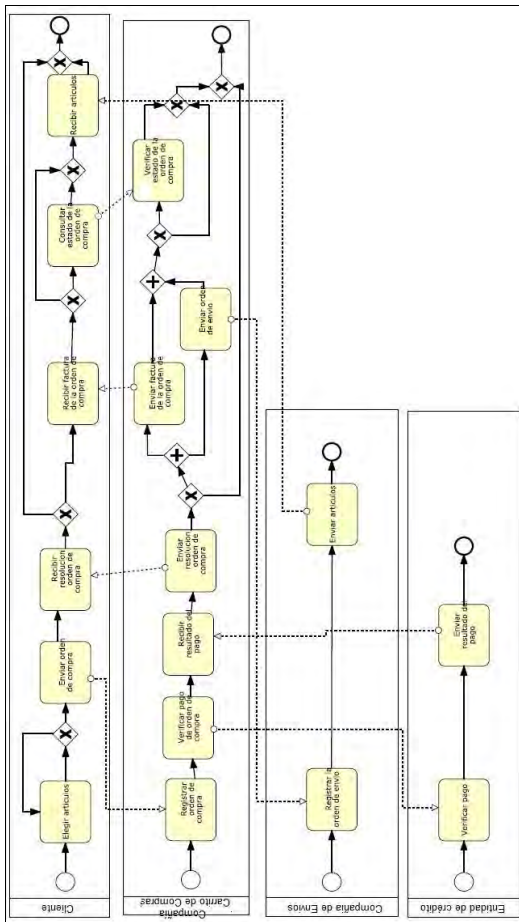
2.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo



MATERIAL EXPERIMENTAL – GRUPO A

Sección 1 – Planteamiento del problema: descripción del Proceso de negocio colaborativo “Carrito de compras” de una Compañía.



Descripción del proceso de negocio “Carrito de Compras”

El Cliente elige en la página Web de la Compañía los artículos que quiere integrar en el carrito de compras así como los datos de pago y dirección de envío asociada, y confirma la orden de compra correspondiente.

La Compañía Carrito de compras registra la orden de compra recibida y envía el pago asociado a la Entidad de Crédito correspondiente, de la cual recibe el resultado correspondiente al pago que a su vez envía al Cliente asociado a la orden de compra registrada. Si el pago fue exitoso entonces envía a la Compañía de Emisos la orden de envío asociada con los datos del Cliente registrados, y ésta realiza la entrega de los artículos correspondientes. En cualquier momento el Cliente puede chequear el estado de la orden de compra en la página Web de la Compañía. Si el pago no fue exitoso el proceso de negocio termina, pudiendo el Cliente si así lo desea realizar un nuevo pedido, lo cual no está modelado en el proceso.

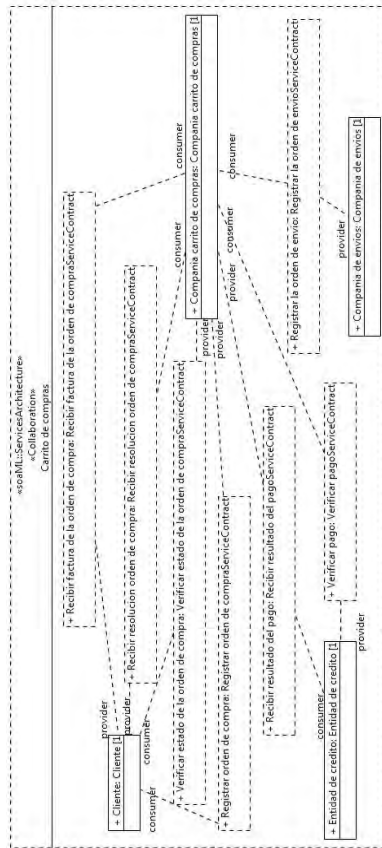
Sección 2 – Definición de servicios para la implementación del proceso de negocio descrito en la Sección 1 “Carrito de compras” de una Compañía. Diseño de servicios en SoaML.

En esta sección se brindan diversos diagramas especificados en SoaML para el modelado de los servicios que considera necesarios para implementar el proceso de negocio presentado en la Sección 1.

En cada pregunta se plantean 6 opciones relacionadas con el modelo de servicios que se presenta que ud. deberá responder con S/NO, y una pregunta sobre la COMPLEJIDAD del modelo presentado, en la escala de 1 a 5 siendo: 1 – muy simple, 2 – algo simple, 3 – normal, 4 – algo complejo, 5 – muy complejo

1) Arquitectura de Servicios

Esta pregunta requiere identificar elementos relacionados con la Arquitectura de servicios definida para realizar el proceso de negocio “Carrito de compras”.



1.1) El rol provider en el contrato de servicio del servicio “Registrar orden de compra” es jugado por el participante Compañía carrito de compras. V F

1.2) El participante Cliente provee los servicios “Recibir factura de la orden de compra”, “Recibir resolución orden de compra”, “Verificar estado de la orden de compra”, “Registrar orden de compra”. V F

1.3) El participante Entidad de crédito juega el rol consumer en el contrato de servicio del servicio “Verificar pago”. V F

1.4) El contrato de servicio para el servicio “Recibir resultado del pago” define la interacción de los participantes Compañía carrito de compras y Entidad de crédito con el servicio con los roles provider y consumer respectivamente. V F

1.5) El participante Compañía de emisos interactúa con el participante Entidad de crédito en el servicio “Registrar la orden de envío” con el rol provider. V F

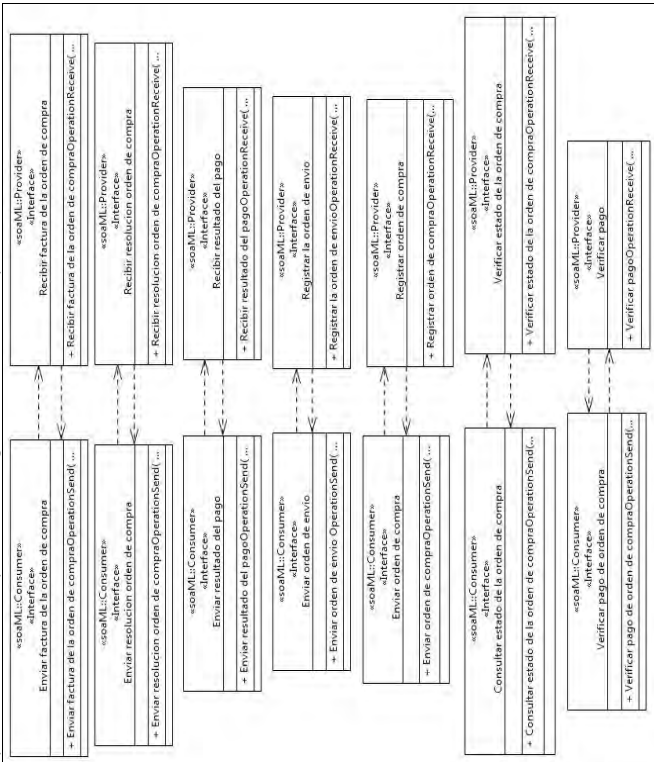
1.6) Los roles definidos para el servicio “Verificar estado de la orden de compra” consumer y provider son jugados por los participantes Cliente y Compañía carrito de compras respectivamente. V F

1.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo

2) Interfaces de servicios

Esta pregunta requiere identificar elementos relacionados con las interfaces provistas y requeridas definidas para los servicios del proceso de negocio "Carrito de compras".

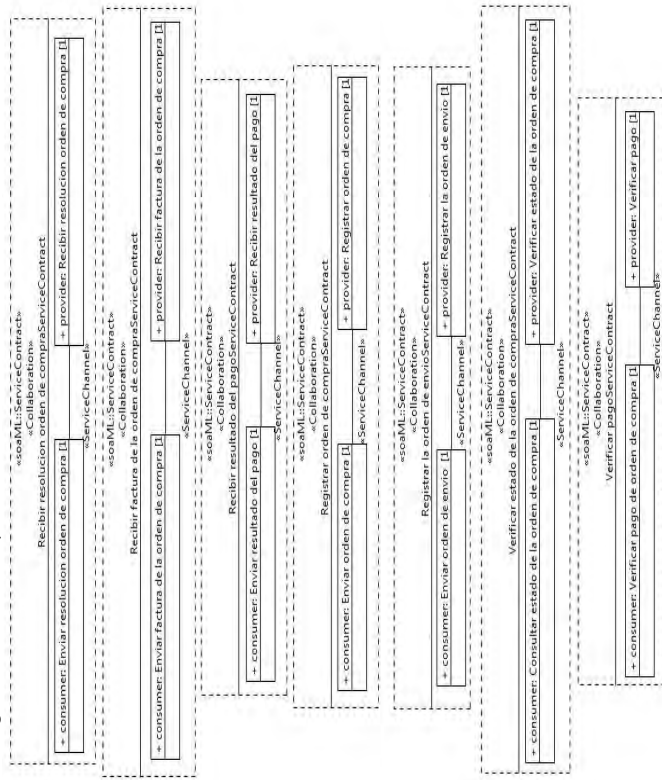


- 2.1) "Consultar estado de la orden de compra" es la interface provista para ser invocada cuando se quiere obtener el estado actual de la orden de compra ingresada. V F
- 2.2) La interface "Registrar la orden de envío" provee la operación "Registrar la orden de envíoOperationReceive" que es invocada por la interface "Enviar orden de envío". V F
- 2.3) El servicio "Recibir resultado del pago" se define con las interfaces "Recibir resultado del pago" y "Enviar resultado del pago", que se invocan entre sí mediante las operaciones definidas en cada interface. V F
- 2.4) La interface "Verificar pago de orden de compra" provee la operación "Verificar pago de orden de compraOperation Send" que es invocada por el proveedor para iniciar la interacción con el servicio "Verificar pago". V F
- 2.5) La interface "Registrar orden de compra" invoca la operación "Enviar orden de compraOperation Send" para contestar en la interacción definida para el servicio. V F
- 2.6) La interface "Enviar resultado del pago" es invocada por la Compañía carrito de compras. V F
- 2.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo

3) Contratos de servicios

Esta pregunta requiere identificar elementos relacionados con los contratos definidos para los servicios del proceso de negocio "Carrito de compras".



- 3.1) El rol consumer del servicio "Registrar la orden de envío" es de tipo Enviar orden de envío que corresponde con la interface provista para ingresar la orden de envío. V F
- 3.2) Los roles consumer y provider definidos en el contrato de servicio para el servicio "Registrar orden de compra" por la Compañía carrito de compras son del tipo definido para las interfaces del servicio respectivamente. V F
- 3.3) El servicio "Recibir resolución orden de compra" provee el rol provider para que la interface "Enviar resolución orden de compra" sea la interface del proveedor para la interacción. V F
- 3.4) La interface "Enviar resultado del pago" que es el tipo del rol consumer en el contrato del servicio "Recibir resultado del pago" es invocada por la interface en el rol provider para responder al envío de información realizado. V F
- 3.5) Los roles definidos para el servicio "Verificar estado de la orden de compra" son consumer del tipo de la interface "Consultar estado de la orden de compra" y provider del tipo de la interface "Verificar estado de la orden de compra". V F
- 3.6) El rol provider en el contrato de servicio del servicio "Recibir factura de la orden de compra" es de tipo "Recibir factura de la orden de compra" que corresponde con la interface provider definida para el servicio. V F
- 3.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo





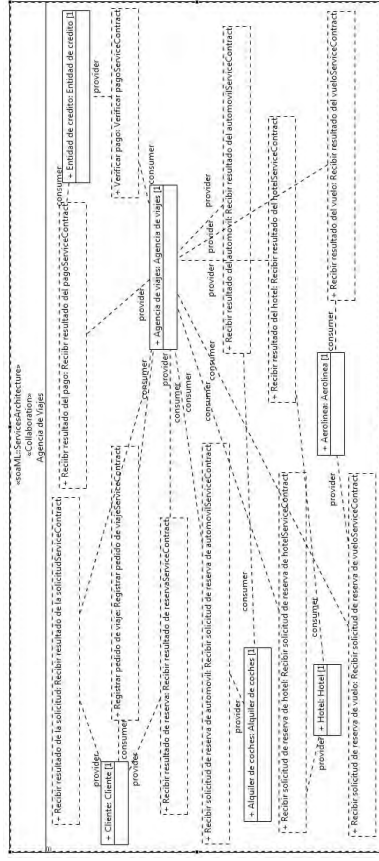
**Sección 2 – Definición de servicios para la implementación del proceso de negocio descrito en la Sección 1 “Reserva de viaje” de una Agencia de viajes. Diseño de servicios en SoaML.**

En esta sección se brindan diversos diagramas especificados en SoaML para el modelado de los servicios que considera necesarios para implementar el proceso de negocio presentado en la Sección 1.

En cada pregunta se plantean 6 opciones relacionadas con el modelo de servicios que se presenta que ud. deberá responder con SI/NO, y una pregunta sobre la COMPLEJIDAD del modelo presentado, en la escala de 1 a 5 siendo: 1 – muy simple, 2 – algo simple, 3 – normal, 4 – algo complejo, 5 – muy complejo

**1) Arquitectura de Servicios**

Esta pregunta requiere identificar elementos relacionados con la Arquitectura de servicios definida para realizar el proceso de negocio “Reserva de viaje”.



- 1.1) El rol proveedor en el contrato de servicio del servicio “Recibir solicitud de reserva de vuelo” es jugado por el participante Aerolineas. V F
- 1.2) El participante Agencia de viajes provee los servicios “Registrar pedido de viaje”, “Recibir resultado de pago”, “Recibir resultado de automóvil”, “Recibir resultado del hotel”, “Recibir resultado del vuelo”. V F
- 1.3) El contrato de servicio para el servicio “Recibir resultado de la solicitud” define la interacción de los participantes Agencia de viajes y Cliente con el servicio con los roles consumer y provider respectivamente. V F
- 1.4) El participante Cliente juega el rol consumer definido en el contrato de servicio del servicio “Recibir resultado de reserva”. V F
- 1.5) El participante Hotel interactúa con el participante Cliente en el servicio “Recibir resultado del hotel” con el rol consumer. V F
- 1.6) Los roles definidos para el servicio “Verificar pago” consumer y provider son jugados por los participantes Agencia de viajes y Entidad de crédito respectivamente. V F
- 1.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo

**Descripción del proceso de negocio “Reserva de viaje”**

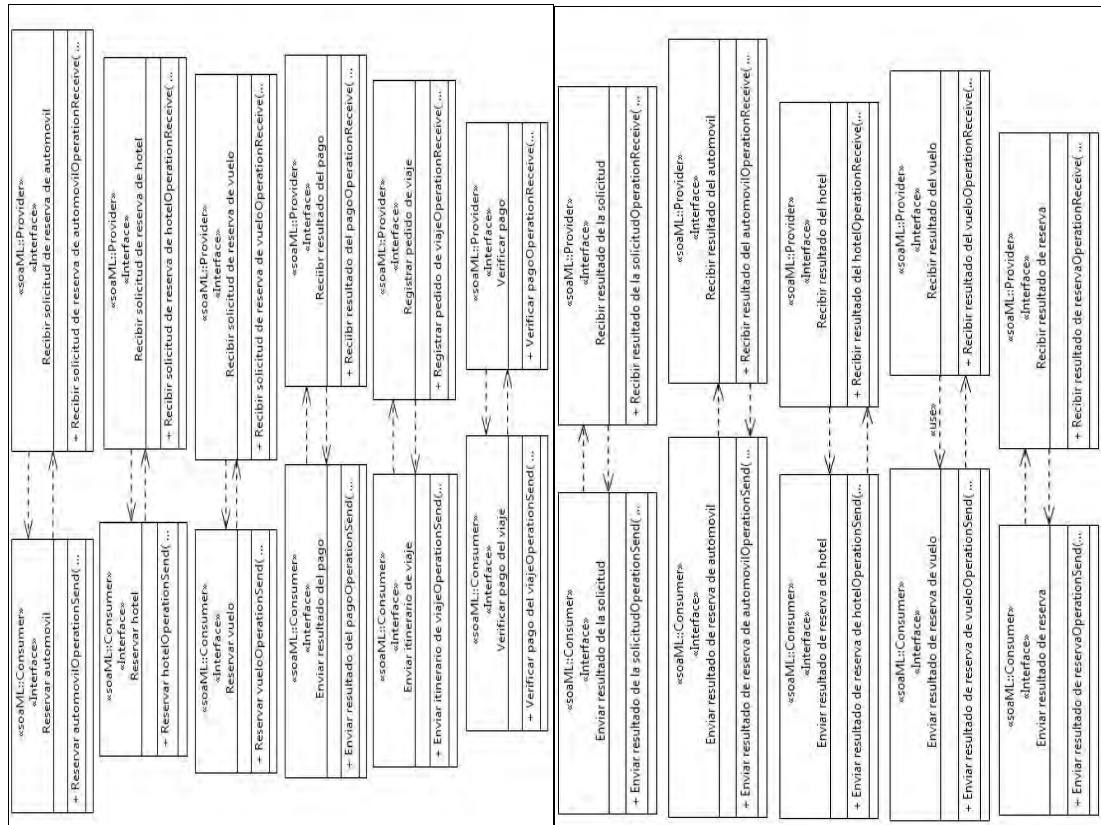
El cliente ingresa una solicitud de Reserva de viaje en el sitio Web de la Agencia de viajes, indicando fechas de vuelos, cantidad de días de hotel y reserva de automóvil de alquiler, en cualquier combinación de las anteriores, así como la forma de pago (tarjeta de crédito, transferencia bancaria, etc. indicando la información necesaria en cada opción).

La Agencia de Viajes confirma el pago con la Entidad de crédito y envía el resultado de la solicitud al cliente, en el caso que no se haya podido cargar el importe el Cliente tiene la posibilidad de cambiar de opción de pago. Si el cargo fue confirmado la Agencia de Viajes solicita entonces las reservas a las entidades correspondientes: Aerolínea, Hotel y Alquiler de coche, las cuales envían el resultado de las mismas a la Agencia de viajes.

La Agencia de viajes registra el resultado de la reserva según las contestaciones recibidas, y envía el resultado de la reserva al Cliente. En el caso que no se haya podido realizar alguna reserva el Cliente tiene la opción de ingresar nuevamente a realizar otra solicitud o modificar la solicitud con reservas pendientes, lo cual no está modelado en el diagrama.

2) Interfaces de servicios

Esta pregunta requiere identificar elementos relacionados con las interfaces provistas y requeridas definidas para los servicios del proceso de negocio "Reserva de viaje".



2.1) La interface "Enviar itinerario de viaje" provee la operación "Registrar pedido de viaje". V F

2.2) "Reservar automóvil" es la interface provista para ser invocada cuando se quiere reservar un automóvil para el viaje. V F

2.3) La interface "Enviar resultado de reserva del hotel" provee la operación "Enviar resultado de reserva de hotelOperation Send" que es invocada por el proveedor para iniciar la interacción con el servicio "Recibir resultado del hotel". V F

2.4) La interface "Enviar resultado del pago" es invocada por la Entidad de crédito para enviar la información del pago solicitada por la Agencia de viajes. V F

2.5) El servicio "Recibir solicitud de reserva de vuelo" se define con las interfaces "Recibir solicitud de reserva de vuelo" y "Reservar vuelo", que se invocan entre sí mediante las operaciones definidas en cada interface. V F

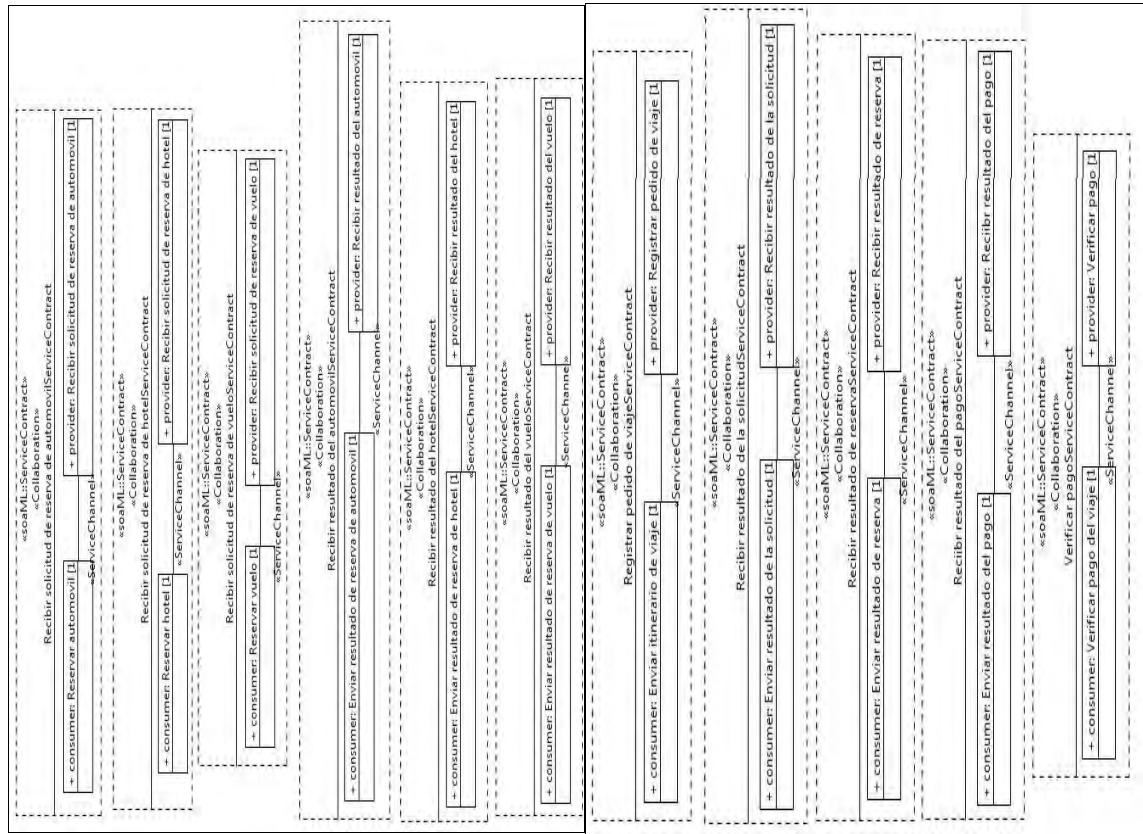
2.6) La interface "Verificar pago" invoca la operación "Verificar pago del viajeOperation Send" para contestar en la interacción definida para el servicio. V F

2.7) Valore la complejidad del modelo presentado:

1 - m  imple 2 - alg  imple 3 -  rmal 4 - algo  plejo 5 - muy c  plejo

**3) Contratos de servicios**

Esta pregunta requiere identificar elementos relacionados con los contratos definidos para los servicios del proceso de negocio "Reserva de viaje".



**Appendix D. Experimental material of the QVT transformations validation experiment**

3.1) El rol consumer del servicio "Registrar pedido de viaje" es de tipo Enviar itinerario de viaje que corresponde con la interface provista para ingresar la reserva del viaje. V F

3.2) El servicio "Recibir resultado de la solicitud" provee el rol provider para que la interface "Enviar resultado de la solicitud" sea la interface del proveedor para la interacción. V F

3.3) Los roles consumer y provider definidos en el contrato de servicio para el servicio "Recibir resultado del automóvil" por la Agencia de viajes son del tipo definido para las interfaces del servicio respectivamente. V F

3.4) Los roles definidos para el servicio "Verificar pago" son consumer del tipo de la interface "Verificar pago del viaje" y provider del tipo de la interface "Verificar pago". V F

3.5) El rol provider en el contrato de servicio del servicio "Recibir resultado de reserva" es de tipo "Recibir resultado de reserva" que corresponde con la interface provider definida para el servicio. V F

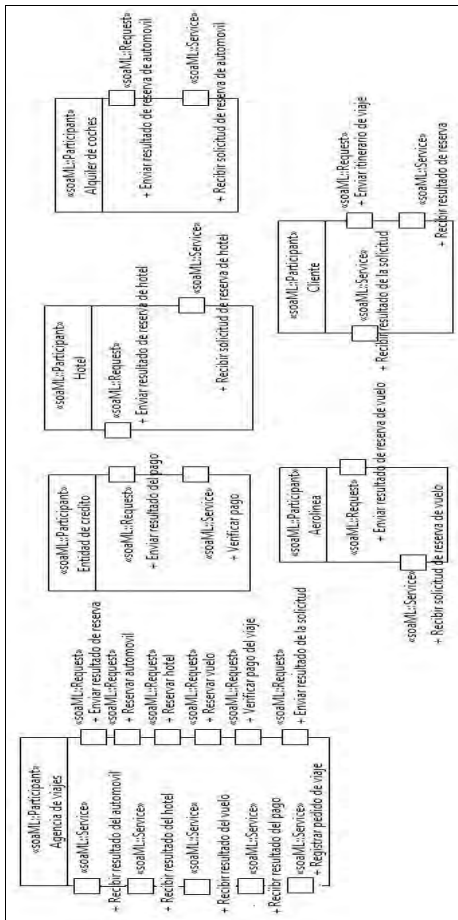
3.6) La interface "Recibir resultado de reserva de hotel" que es el tipo del rol consumer en el contrato del servicio "Recibir resultado de reserva de hotel" es invocada por la interface en el rol provider para responder al envío de información realizado. V F

3.7) Valore la complejidad del modelo presentado:

1 -  simple    2 -  simple    3 -  normal    4 -  complejo    5 -  muy complejo

4) Participantes y servicios asociados

Esta pregunta requiere identificar elementos relacionados con los participantes del proceso de negocio “Reserva de viaje” que proveen y/o consumen servicios.



4.1) El participante Aerolínea provee un servicio para “Recibir solicitud de reserva de vuelo” como indican los puertos “Recibir solicitud de reserva de vuelo” y “Enviar resultado de reserva de vuelo” en el participante. V F

4.2) Los puertos “Recibir resultado de la solicitud”, y “Recibir resultado de reserva” definen los servicios que el participante Cliente provee. V F

4.3) El participante Agencia de viajes y el participante Entidad de crédito interactúan por medio de dos servicios para: “Verificar pago” del viaje solicitado y “Recibir resultado del pago”, según los puertos definidos en ambos participantes. V F

4.4) El participante Cliente y el participante Hotel no tienen servicios definidos para interactuar entre sí. V F

4.5) El participante Agencia de viajes provee servicios para “Registrar pedido de viaje”, “Reservar automóvil”, “Reservar hotel”, “Reservar vuelo” y “Recibir resultado del pago” según los puertos definidos en el participante. V F

4.6) El participante Cliente y el participante Agencia de viajes interactúan en el servicio “Registrar pedido de viaje” según los puertos “Registrar pedido de viaje” y “Enviar itinerario de viaje” definidos en ambos participantes. V F

4.7) Valore la complejidad del modelo presentado:

1 – muy simple  2 – algo simple  3 – normal  4 – algo complejo  5 – muy complejo



## Appendix E.

# HGCR case study implementation in XPDL and WS-BPEL and simulation of resources

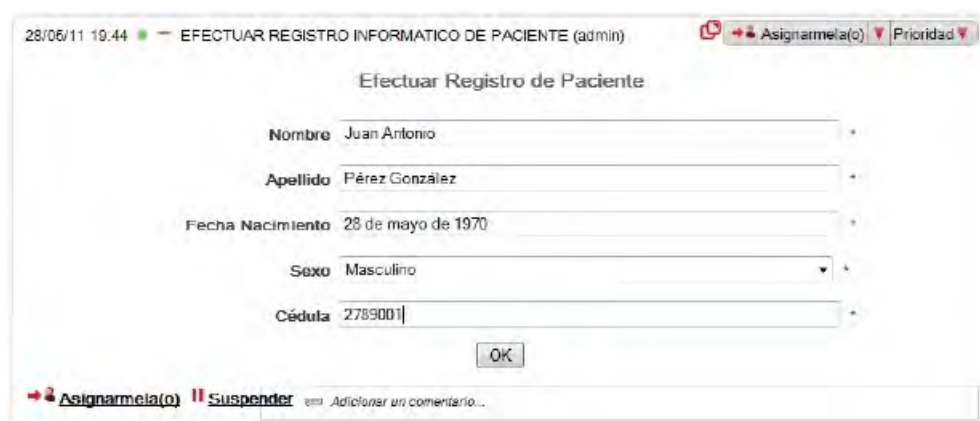
### E.1. Overview

This Appendix complements the implementation and execution of the Patient MAS BP presented in chapter 10 presenting the realization of the case study in the other two selected process engines: Bonita for XPDL and Intalio community edition for WS-BPEL. In Bonita the same sub-processes as presented for Activiti, “Admission and Registration and Preparation for MAS” were used, and for Intalio the “Pre-intervention” sub-process was selected. In the last section, another alternative for modeling resources for the CPNTools simulation than the one presented in chapter 10 is discussed.

### E.2. Implementation and execution in Bonita

The “Patient Admission and Registration” and “Preparation for MAS” sub-processes from the Patient MAS BP were modeled in Bonita and deployed to the process engine to be executed, as Bonita is an integrated suite which provides all tools in one. It allows modeling pools so the BP was specified as defined in the original one from the HGCR, which is shown in Figure E.2. Bonita does not allow direct Message flows from one pool to another, which has to be modeled with an intermediate message element from the sender, as shown.

For the user activities the corresponding user forms were defined, from which an example is shown in Figure E.1 for the registration of the patient, with fields name, last name, birth date, sex and document ID.



The screenshot shows a web-based user form for patient registration. The title is "Efectuar Registro de Paciente". The form contains the following fields and values:

- Nombre: Juan Antonio
- Apellido: Pérez González
- Fecha Nacimiento: 28 de mayo de 1970
- Sexo: Masculino
- Cédula: 2789001

At the bottom of the form, there is an "OK" button and a toolbar with icons for "Asignar tarea(s)", "Suspender", and "Añadir un comentario...". The top of the window shows the date and time "28/06/11 10:44" and the user role "EFFECTUAR REGISTRO INFORMATICO DE PACIENTE (admin)".

Figure E.1.: Example of users forms defined in Bonita

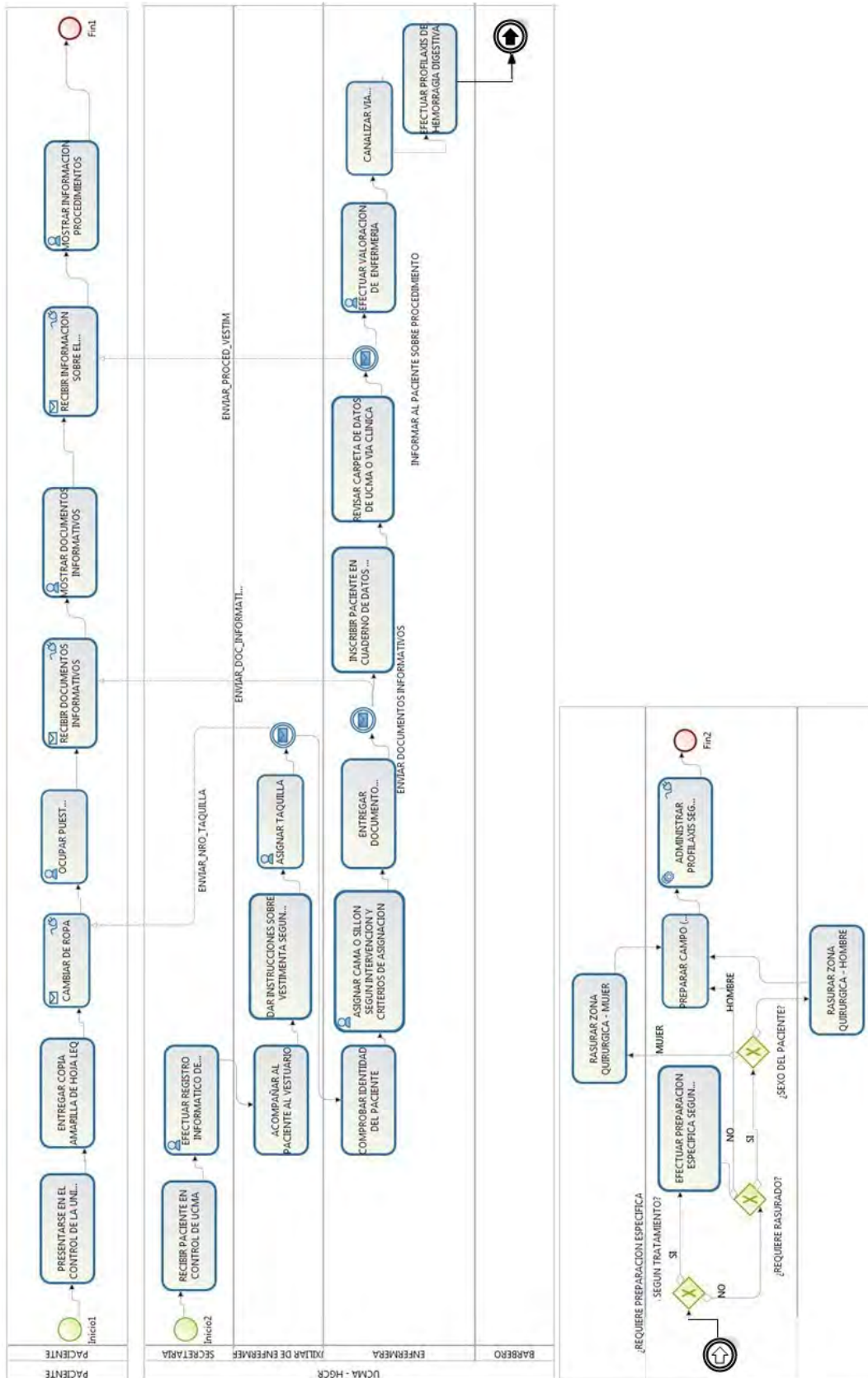


Figure E.2.: Patient MAS BP sub-processes modeled in Bonita



Each of the participants has the corresponding tasks list from which can choose to perform the associated activities, which is shown in Figure E.3 for the activity of assigning the number of locker for the clothes of the patient.



Figure E.3.: Example of tasks list in Bonita

For the ServiceTask implementation the sending of an email was also performed from the Bonita process engine, which is shown in Figure E.4.

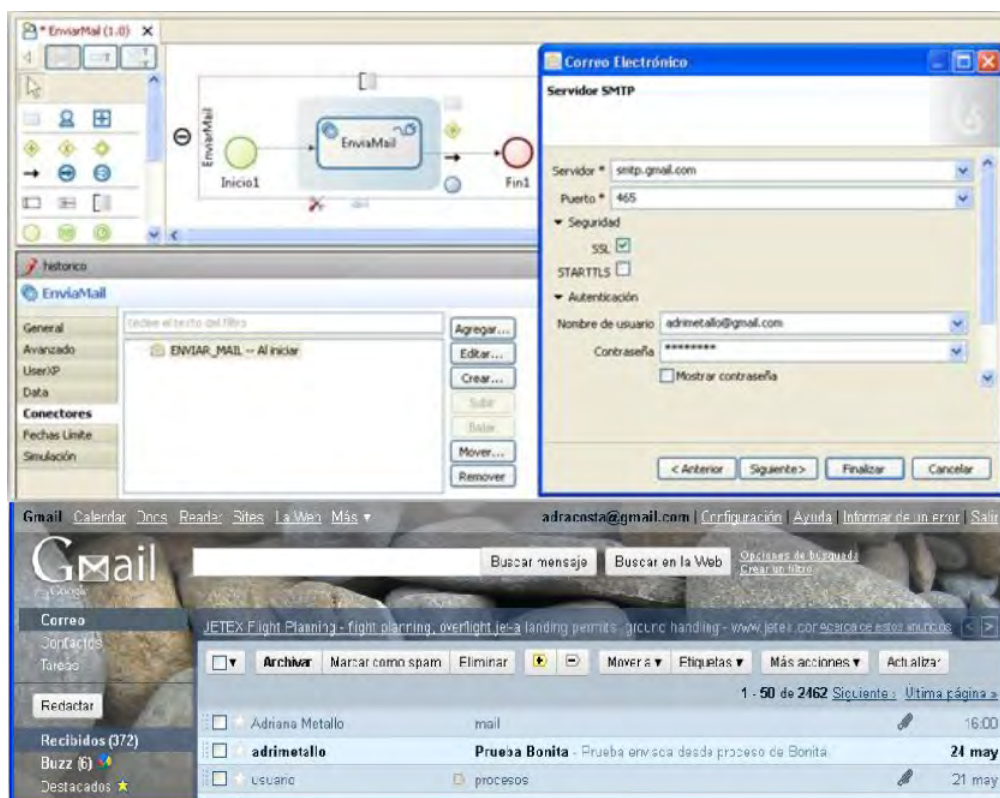


Figure E.4.: Example of ServiceTask as sending email in Bonita

After the Patient MAS BP has been executed several times, the registered data about the BP execution was gathered from the data base of the Bonita process engine, to be transformed in MXML format to be loaded in the ProM framework. Bonita also has a predefined data base that can be changed to several existing ones such as MySQL or Postgress. For this case study the MySQL data base was selected and a query example is shown in Figure E.5.

```

SELECT p.DBID_PROC_NRO, p.NAME_PROC_NOMBRE,
       p.STATE_PROC_ESTADO, pi.NB_PROC_INSTANCIA,
       from_unixtime((pi.START_DATE/1000)) PROC_INST_INI,
       from_unixtime((pi.END_DATE/1000)) PROC_INST_FIN,
       pi.START_BY PROC_USUARIO_INI, pi.END_BY PROC_USUARIO_FIN,
       pi.INST_STATE_PROC_INST_ESTADO,
       a.DBID_ACT_NRO, a.LABEL_ACT_NOMBRE,
       ai.activity_state ACT_INST_ESTADO,
       from_unixtime((ai.READY_DATE/1000)) ACT_INST_ACTIVIA,
       from_unixtime((ai.START_DATE/1000)) ACT_INST_INI,
       from_unixtime((ai.END_DATE/1000)) ACT_INST_FIN,
       ai.START_BY ACT_USUARIO_INI, ai.END_BY ACT_USUARIO_FIN,
       ai.ACTIVITY_STATE_ACT_INST_ESTADO, ai.TYPE_ACT_TIPO
FROM bonita_core.bn_proc_def p,
     bonita_history.bn_proc_inst pi,
     bonita_core.bn_act_def a,
     bonita_history.bn_act_inst ai
where p.proc_uid_ = pi.process_uid_
and pi.process_uid_ = a.process_uid_
and a.process_uid_ = ai.process_uid_
and a.act_uid_ = ai.act_def_uid_
and ai.instance_dbid = pi.dbid
order by pi.NB_, ai.START_DATE
    
```

Figure E.5.: Example of queries from Bonita data base

The transformation of the .csv file into MXML format performed in Fluxicon is shown in Figure E.6, where the it can be seen from left to right, the row number, the number of the BP, the name of the BP corresponding to the defined pools, the state of the BP, the number of the BP instance, the data and hour of start of the BP instance, the data and hour of the completion of the BP instance, the associated execution user initiating and completing the BP instance. The rest of the information corresponding to the activities instances data is not shown due to space issues for taking the screenshot.

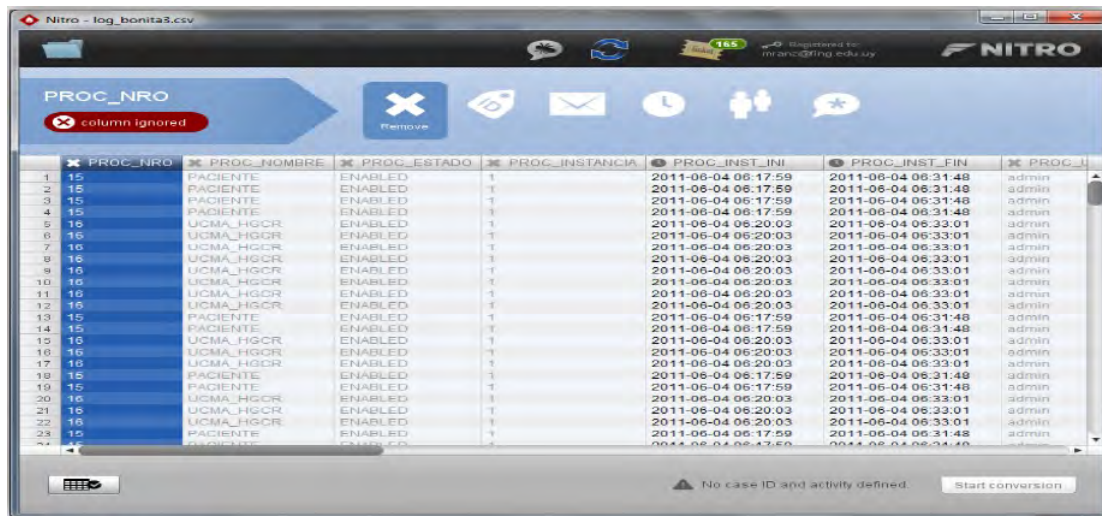


Figure E.6.: Fluxicon transformation to MXML format from Bonita execution

Once the event log is transformed into the MXML format it can be loaded in ProM for further analysis, which is shown in Figure E.7.



Figure E.7.: Bonita MXML log loaded into ProM

### E.3. Implementation and execution in Intalio

The Pre-intervention sub-process from the Patient MAS BP was modeled in Intalio designer, which is shown in Figure E.9. Intalio allows to model several pools but only one is executed and the others are modeled for showing interactions with user forms and with Web Services, in this case the active pool is the one corresponding to the HGCR in the center. The other two pools marked as inactive with dark grey, correspond to: the one on the top to the interaction with the user forms defined for registering the patient, and for registering the information on the anesthesia; and the one on the bottom corresponds to the interaction with a defined Web Service to register the patient in the operation room book.

For the user activities the corresponding user forms were defined, from which an example is shown in Figure E.8 for performing the pre-intervention evaluation of the patient, such as previous illness, previous surgeries, physical exams carried out, and the final recommendation about the anesthesia to be given to the patient and the patient risk assessment.

**INTALIO**

Tasks | Notifications | Processes

**Enfermedades Anteriores**

- Enf. Pulmonar
- Enf. Riñon
- Enf. Corazon
- Enf. Hepática
- Alergias

**Antecedentes**

- Intervenciones anteriores
- Problemas anestésicos
- Problemas familiares anestésicos

**Exámenes realizados**

Coagulación: 6 minutos

Htes: 140/90      Hto: 25%      Hb: 44%

**Conclusiones**

1- Anestesia recomendada:  Local,  General

2- Riesgo elevado:  SI,  NO

Buttons: Claim, Save, Complete

Figure E.8.: Example user forms defined in Intalio

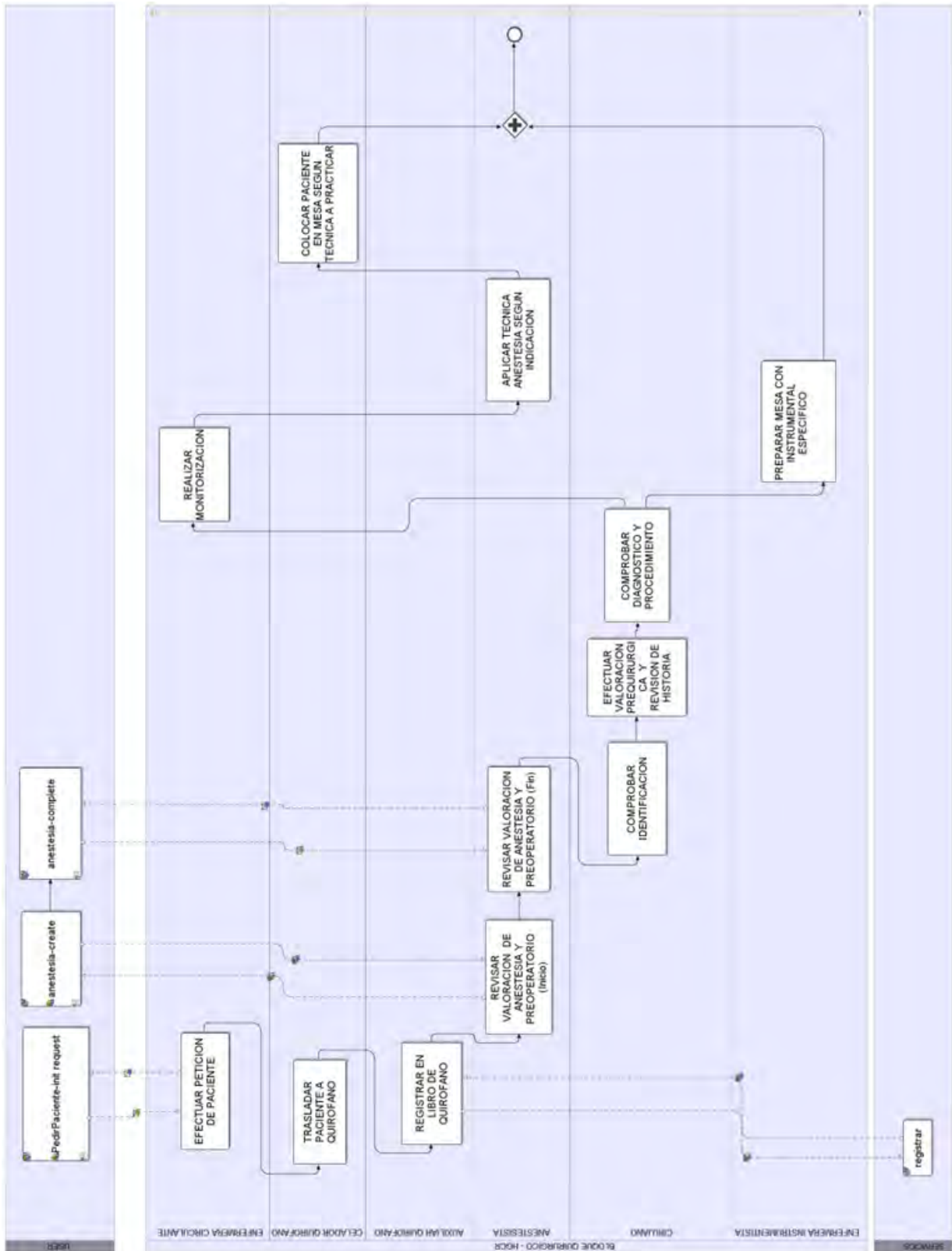


Figure E.9.: Pre-intervention sub-process modeled in Intalio designer



Timestamp	Event Name	Activity Lifecycle	Instance Lifecycle	Scope Name
2011-07-18 06:12:49	ActivityLifecycleEvent	activityLifecycle	100 250 500 750 1000	REGISTRAR_EN_LIBRO_DE_QUIROFANO-1
2011-07-18 06:12:49PM	ActivityExecStartEvent	activityLifecycle		REGISTRAR_EN_LIBRO_DE_QUIROFANO-1
2011-07-18 06:12:49PM	VariableReadEvent	dataHandling		
2011-07-18 06:12:50PM	ProcessMessageExchangeEvent	instanceLifecycle		
2011-07-18 06:12:50PM	ProcessMessageExchangeEvent	instanceLifecycle		
2011-07-18 06:12:50PM	VariableModificationEvent	dataHandling		
2011-07-18 06:12:50PM	ActivityExecEndEvent	activityLifecycle		REGISTRAR_EN_LIBRO_DE_QUIROFANO-1
2011-07-18 06:12:50PM	ActivityExecEndEvent	activityLifecycle		
2011-07-18 06:12:50PM	ActivityEnabledEvent	activityLifecycle	assign-activity-line-256	
2011-07-18 06:12:50PM	ActivityExecStartEvent	activityLifecycle	assign-activity-line-256	
2011-07-18 06:12:50PM	VariableModificationEvent	dataHandling		

parent-scope-id	1998850
process-id	blog:BLOQUE QUIRURGICO_-_HGCR-17
name	VariableModificationEvent
line-number	256
instance-id	1933313
scope-definition-id	99
variable-name	admLibroRegistrarResponseMsg
timestamp	2011-07-18T18:12:50.421-03:00
new-value	<?xml version="1.0" encoding="UTF-8"?> <message><parameters><registrarResponse xmlns="http://quirofano. xmlns.ns="http://quirofano. xmlns.soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <return>OK</return> </registrarResponse></parameters><Action headerPart="true"><Action xmlns="http://www.w3.org/2005/08/addressing" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsa="http://www.w3.org/2005/08/addressing"><RelatesTo xmlns="http://www.w3.org/2005/08/addressing" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsa="http://www.w3.org/2005/08/addressing">uid.0e4bd716-ed4e-420a-b0e9-f6ecfba3269-5</RelatesTo></Action></message>
type	dataHandling
scope-id	1998851
scope-name	REGISTRAR_EN_LIBRO_DE_QUIROFANO-1

Figure E.12.: Web service message response in Intalio

After several executions of the BP the data registered is extracted from the Intalio data base, which is simpler than the ones presented before so the query is performed on one table as “Select \* from bpel\_event where PID= <id\_proceso>,” which returns all the events registered for each executed BP as <ID,TSTAMP,TYPE,DETAIL>, where ID corresponds to the executed instance, TSTAMP is the timestamp corresponding to the registered event, TYPE is the event type (start, complete, among others) and DETAIL is a string containing extra data. The .csv file obtained is loaded into ProMImport framework to be transformed into MXML format to be loaded in ProM, which is shown in Figure E.13.

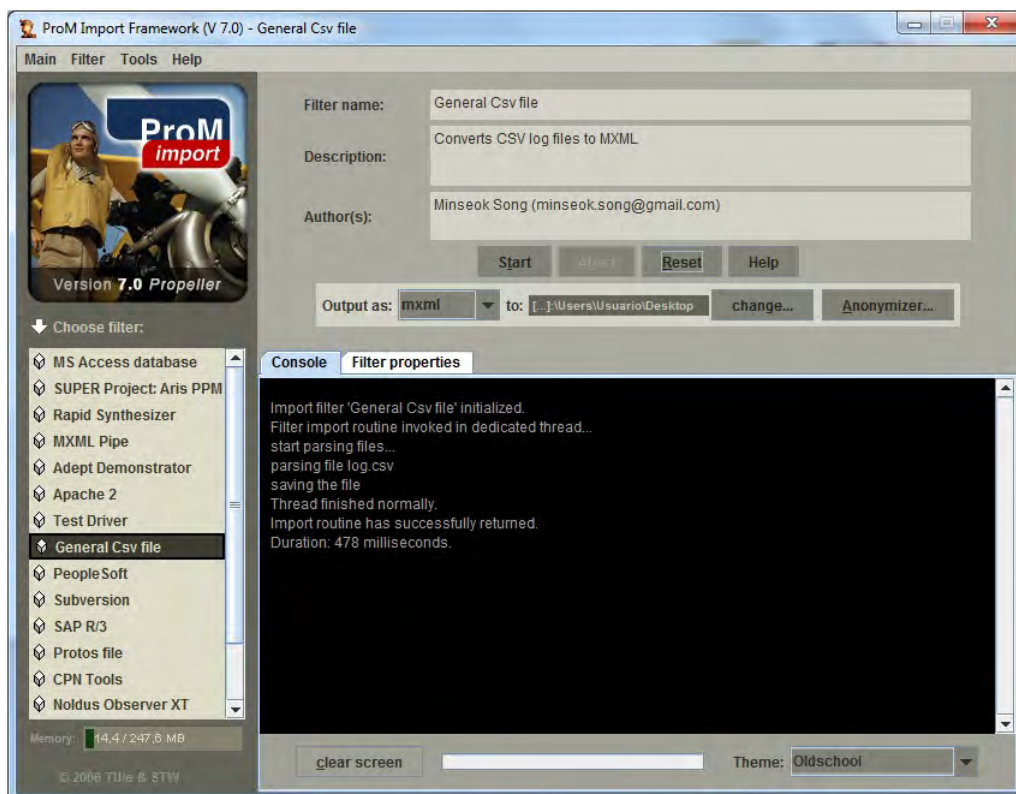


Figure E.13.: ProMImport framework for transforming .csv file into MXML

After transforming the .csv file containing the execution data registered for the BP, the output MXML file is imported into the ProM framework which is shown in Figure E.14.

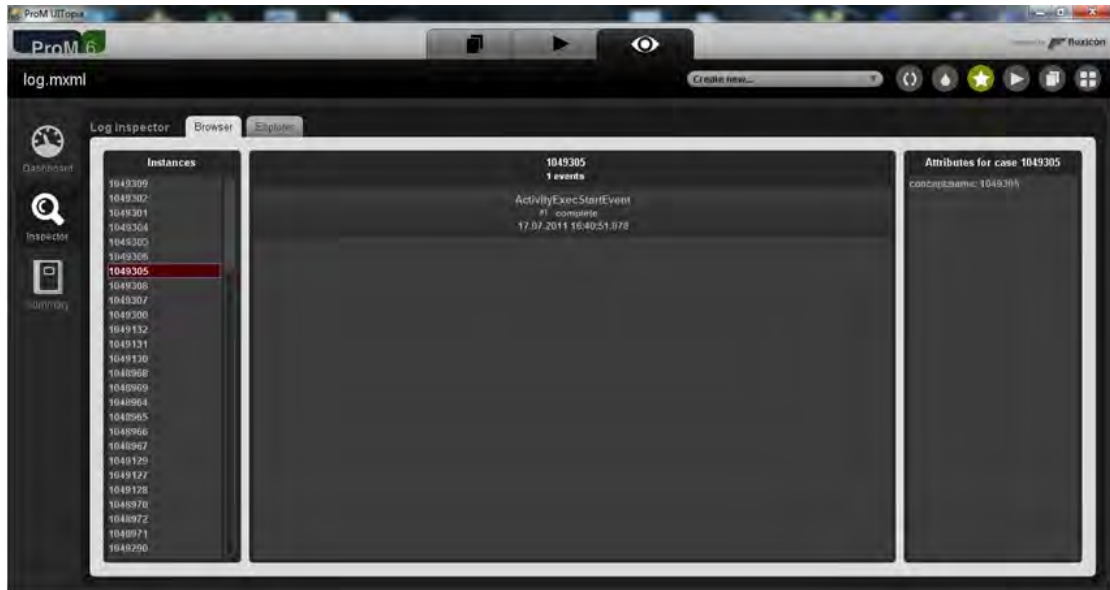


Figure E.14.: Intalio MXML file loaded into ProM framework

## E.4. Alternative modeling of resources in CPN Tools

We tried the resources modeling approach for simulation defined in [van der Aalst et al., 2010], where it is stated that people do not work in an activity from it start to its completion, but in “chunks” of time, suspending and resuming the execution of the activity and changing from one BP to another during their work hours in a day and between different days.

This approach was combined with the one of a central fusion place for resources we were using as presented in chapter 10, as we believe is even more realistic for the simulation of BPs execution, but the complexity for the modeling of each activity increased, as an Activation sub-page has to be added to each activity where the fusion places for the resources are located. In this way resources are not only divided between BP cases and activities instances as they are in the approach we have applied, but also into chunks of work corresponding to defined periods of allowed continuous work.

We finally decided to use the approach of a central fusion place as presented in chapter 10, without adding the time consideration, as it was enough for our purpose in this case study; nevertheless, the modeling of the second approach for the same activity Check preconditions for MAS is shown in Figure E.15 as a comparative example, and the corresponding activation page in Figure E.16, where the increase in complexity can be seen.

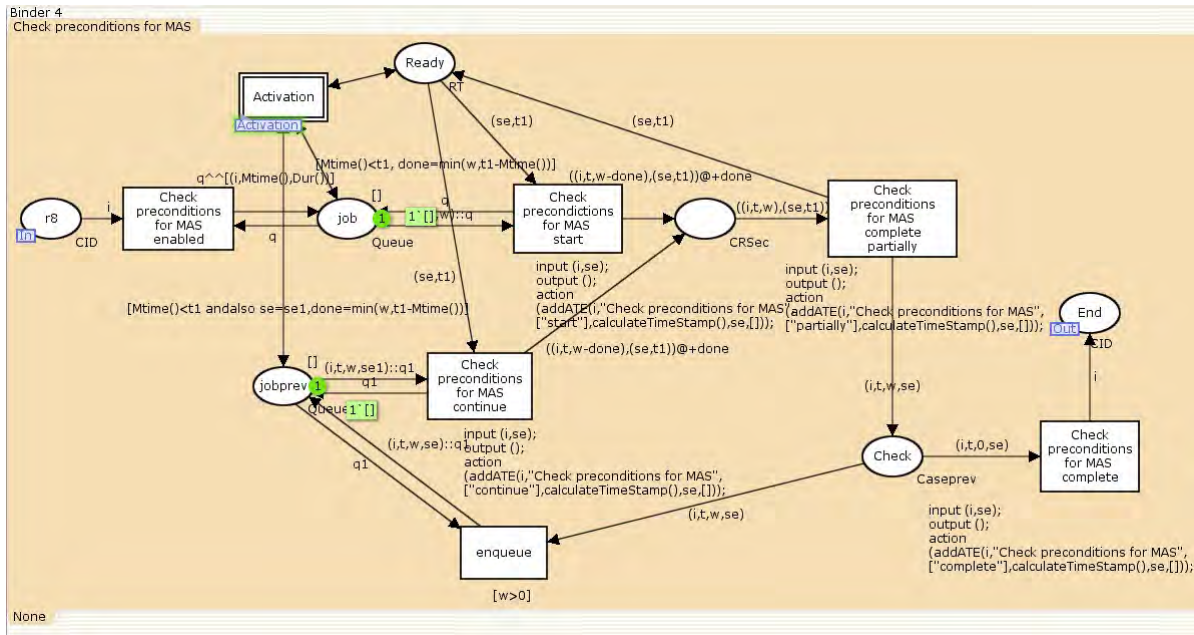


Figure E.15.: Resources modeling with fusion places and “chunks” for Check preconditions for MAS

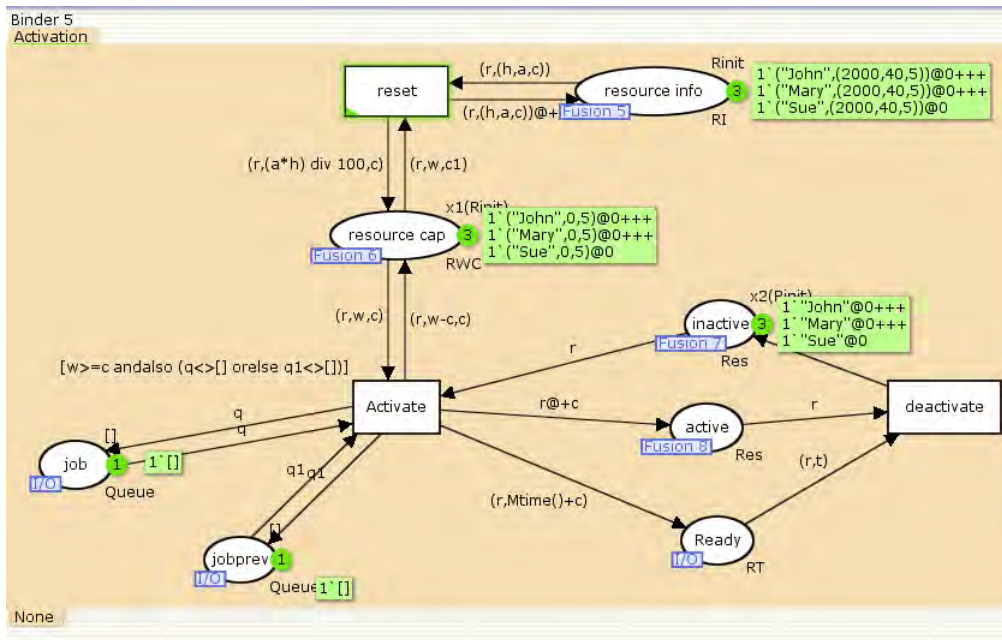


Figure E.16.: Resources activation page for Check preconditions for MAS



# Bibliography

- Alarcos. Empirical-webgen. <http://webgen.webportalquality.com/>, 2006.
- A. Alves de Medeiros and C. Guenther. Process mining: Using cpn tools to create test logs for mining algorithms. In *6th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2005.
- D. Avison, F. Lan, M. Myers, and A. Nielsen. Action research. *Communications of the ACM*, 42 Issue 1:94–97, 1999.
- E. Babbie. *Survey research methods*. Wadsworth Publishing, 1990.
- Y. Baghdadi. Abba: an architecture for deploying business-to-business electronic commerce applications. *Electronic Commerce Research and Applications*, 3 Issue 2, 2004.
- L. Bai and J. Wei. A service-oriented business process modeling methodology and implementation. In *International Conference on Interoperability for Enterprise Software and Applications (IESA'09)*, 2009.
- K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema. Developing applications using model-driven design environments. *IEEE Computer*, 2006.
- M. Barbacci, M. Klein, T. Longstaff, and C. Weinstock. Quality attributes. Technical report, Software Engineering Institute (SEI), CMU/SEI-95-TR-021, 1995.
- V. Basili. Software modeling and measurement: The gqm paradigm. Technical report, University of Maryland, CS-TR-2956, 1992.
- V. Basili. Using experiments to build a body of knowledge, 7th european workshop on software process technology (ewspt 2000). <http://www.cs.umd.edu/basili/presentations/body.knowledge.austria00.pdf>, 2000.
- V. R. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25, Issue 4:456–473, 1999.
- R. Baskerville. Investigating information systems with action research. *Communications of the Association for Information Systems*, 2, article 19, 1999.
- R. Baskerville and A. T. Wood-Harper. A critical perspective on action research as a method for information systems research. *Information Technology*, 3, Issue 11:235–246, 1996.
- L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison Wesley, 2003.
- J. Bezivin and O. Gerbe. Towards a precise definition of the omg/mda framework. In *Automated Software Engineering (ASE'01)*, 2001.
- BPMI. Business process management initiative. <http://www.bpmi.org/>, 2000-2005.
- M. Brambilla, M. Dosmi, and P. Fraternali. Model-driven engineering of service orchestrations. In *Proceedings of the 2009 IEEE Congress on Services (SERVICES'09)*, 2009.
- N. Brand and H. van der Kolk. Workflow analysis and design, 1995.
- P. Brereton, B. Kitchenham, D. Budgen, and Z. Li. Using a protocol template for case study planning. In *Evaluation and Assessment in Software Engineering (EASE'08)*, 2008.
- L. Briand, S. Morasca, and V. Basili. An operational process for goal-driven definition of measures. *IEEE Transactions on Software Engineering*, 28, Issue 12:1106–1125, 2002.

- 
- T. Bruckmann and V. Grunh. Amabulo- a model architecture for business logic. In *15th IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS'08)*, 2008.
- J. Bézivin. Model engineering for software modernization, guest talk at the 11th iee working conference of reverse engineering, 2004.
- J. Bézivin. On the unification power of models. *Software and Systems Modeling*, 4 Issue 2:171–188, 2005.
- J. Cardoso, A. Sheth, and J. Miller. Workflow quality of service. In *International Conference on Enterprise Integration Modeling Technology (ICEIMT/IEM'02)*, 2002.
- M. Castellanos, A. Alves de Medeiros, J. Mendling, B. Weber, and A. Weijters. *Business Process Intelligence, Handbook of Research on BP Modeling*, chapter Business Process Intelligence, Handbook of Research on BP Modeling, pages 456–480. Information Science Reference (IGI Global), 2009.
- C. Cauvet and G. Guzelian. Business process modeling: a service-oriented approach. In *41st Hawaii International Conference on System Sciences (HICSS'08)*, 2008.
- A. Chen and D. Buchs. A generative business process prototyping framework. In *16th International Workshop on Rapid System Prototyping (RSP'06)*, 2006.
- H. Chen. Towards service engineering: Service orientation and business-it alignment. In *41st Hawaii International Conference on System Sciences (HICSS'08)*, 2008.
- P. Clements, R. Kazman, and M. Klein. *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley, 2001.
- T. D. Cook and D. T. Campbell. *Quasi-experimentation: design and analysis issues for field settings*. Houghton Mifflin, 1979.
- D. CPN Group, University of Aarhus. Cpn tools. <http://cpntools.org/start>.
- K. Dahman, F. Charoy, and C. Godart. Generation of component based architecture from business processes: model driven engineering for soa. In *8th IEEE European Conference on Web Services (ECOWS'10)*, 2010.
- T. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business School Press, 1992.
- T. Davenport and J. Short. The new industrial engineering: Information technology and business process redesign. *Sloan Management Review*, 31 Issue 4:11–27, 1990.
- V. de Castro, E. Marcos, and M. López Sanz. A model driven method for service composition modelling: a case study. *Web Engineering and Technology*, Vol. 2, No. 4, 2006.
- V. de Castro, J. M. Vara Mesa, E. Herrmann, and E. Marcos. A model-driven approach for the alignment of business and information system models. In *9th Mexican International Conference on Computer Science (ENC '08)*, 2008.
- A. Delgado. Metodología de desarrollo para aplicaciones con enfoque service oriented architecture (soa). Master's thesis, Basic Sciences Development Program (PEDECIBA), University of the Republic (UdelaR), 2007. (In spanish only).
- A. Delgado. Bpsom methodology. <http://alarcos.esi.uclm.es/MINERVA/BPSOM/>, 2010-2011.
- A. Delgado. Bpcip continuous improvement process. <http://alarcos.esi.uclm.es/MINERVA/BPCIP/>, 2011.
- A. Delgado and I. García-Rodríguez de Guzmán. Ontología para relacionar procesos de negocio y su realización como servicios. In *II Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS'09) en XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD'09)*, 2009.

- A. Delgado, F. Ruiz, and I. García-Rodríguez de Guzmán. Desarrollo de software orientado a servicios basado en procesos de negocio. In *XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software (IDEAS'09) -from 2010 CibSE Congreso Iberoamericano en "Software Engineering"-*, 2009a.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Towards a service-oriented and model-driven framework with business processes as first-class citizens. In *2nd International Conference on Business Process and Services Computing (BPSC'09)*, 2009b.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Minerva: Model driven and service oriented framework for the continuous business process improvement & related tools. In *5th International Workshop on Engineering Service Oriented Applications (WESOA'09)*, in *(ICSOC'09)*, 2009c.
- A. Delgado, I. García-Rodríguez de Guzmán, F. Ruiz, and M. Piattini. Tool support for service oriented development from business processes. In *2nd. International Workshop on Model-Driven Service Engineering (MoSE'10)*, 2010a.
- A. Delgado, I. García-Rodríguez de Guzmán, F. Ruiz, and M. Piattini. From bpmn business process models to soaml service models: a transformation-driven approach. In *2nd International Conference on Software Technology and Engineering (ICSTE'10)*, 2010b.
- A. Delgado, I. García-Rodríguez de Guzmán, F. Ruiz, and M. Piattini. Generación de modelos de servicios en soaml desde modelos de procesos de negocio en bpmn con qvt. In *VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM'10) en XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD'10)*, 2010c.
- A. Delgado, F. Ruiz, and I. García-Rodríguez de Guzmán. Ontología para el ciclo de vida de los procesos de negocio implementados con servicios. In *Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'10)*, 2010d.
- A. Delgado, F. Ruiz, and I. García-Rodríguez de Guzmán. Mejora continua de procesos de negocio basada en pmcompetisoft integrando bpmn. In *III Taller sobre Procesos de Negocio e Ingeniería de Servicios (PNIS'10) en XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD'10)*, 2010e.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Towards an ontology for service oriented modeling supporting business processes. In *4th International Conference on Research Challenges in Information Science (RCIS'10)*, 2010f.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Application of service-oriented computing and model-driven development paradigms to business processes: a systematic review. In *5th International Conference on Software and Data Technologies (ICSOFT'10)*, 2010g.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. A model-driven and service-oriented framework for the business process improvement. *Systems Integration*, 1, 2010h.
- A. Delgado, I. García-Rodríguez de Guzmán, and F. Ruiz. Desarrollo de servicios con soaml desde procesos de negocio en bpmn: metodología y automatización. In *VII Jornadas de Ciencia e Ingeniería de Servicios (JCIS'11)*, 2011a.
- A. Delgado, L. González, S. Larroca, A. Pastorini, F. Ruiz, and I. García-Rodríguez de Guzmán. Soaml eclipse plug-in para modelado de servicios, eclipse soaml plug-in demo. In *XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD'11)*, 2011b.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Business process service oriented methodology (bpsom) with service generation in soaml. In *23rd International Conference on Advanced Information Systems Engineering (CAiSE'11)m*, 2011c.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and B. Weber. Marco minerva para mejora continua de procesos de negocio implementados con servicios. In *XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD'11)*, 2011d.

- 
- A. Delgado, B. Weber, F. Ruiz, and I. García-Rodríguez de Guzmán. Execution measurement-driven continuous improvement of business processes implemented by services. In *6th. International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'11)*, 2011e.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Model transformations for business-it alignment: from collaborative business process to soaml service model. In *27th Symposium On Applied Computing (SAC'12)*, 2012a.
- A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. *Main principles on the integration of SOC and MDD paradigms to business processes: a systematic review, Selection of best papers ICSoft 2010*, chapter Main principles on the integration of SOC and MDD paradigms to business processes: a systematic review, pages 88–108. Communications in Computer and Information Science (CCIS) series, 170, Springer-Verlag, 2012b.
- A. Delgado, B. Weber, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. *Continuous improvement of business processes realized by services based on execution measurement, Selection of best papers ENASE 2011*, chapter Continuous improvement of business processes realized by services based on execution measurement, pages 64–81. Communications in Computer and Information Science (CCIS) series, 275, Springer-Verlag, 2012c.
- M. Dumas, W. van der Aalst, and A. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
- Eclipse. Eclipse process framework composer (epf composer). <http://www.eclipse.org/epf/>, 2004–2011.
- B. Elvesaeter, D. Panfilenko, S. Jacobi, and C. Hahn. Aligning business and it models in service-oriented architectures using bpmn and soaml. In *In: 1st International Workshop on Model-Driven Interoperability (MDI'10)*, 2010.
- M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling. *Patterns: Service-Oriented Architecture and Web Services*. IBM Redbooks, 2004.
- T. Erl. *SOA: Concepts, Technology, and Design*. Prentice Hall, 2005.
- R. France and J. Bieman. Multi-view software evolution: A uml-based framework for evolving object-oriented software. In *International Conference on Software Maintenance (ICSM'01)*, 2001.
- W. French and C. Bell. *Organizational Development: Behavioral Science Interventions for Organization Improvement*. Prentice Hall, 1996.
- V. Gacitua-Decar and C. Pahl. Pattern-based business-driven analysis and design of service architectures. In *3rd International Conference on Software and Data Technologies (ICSOFT'08)*, 2008.
- E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
- F. García, M. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, and M. Genero. Towards a consistent terminology for software measurement. *Information and Software Technology*, 48, 2005.
- F. Garcia, F. Ruiz, C. Calero, M. Bertoa, A. Vallecillo, B. Mora, and M. Piattini. Effective use of ontologies in software measurement. *Knowledge Engineering Review*, 1:23–40, 2009.
- D. Garlan and M. Shaw. An introduction to software architecture. Technical report, School of Computer Science, Carnegie Mellon University, CMU-CS-94-166, 1994.
- G. Gartner. Survey analysis: Bpm spending to grow significantly in 2011. <http://www.gartner.com>, February 2011.
- S. K. Greenfield, J. Software factories assembling applications with patterns, models, frameworks and tools. In *Object-Oriented Programming, Systems, Languages & Applications (OOPSLA'03)*, 2003.

- T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, Issue 2, 1993.
- T. Gruber. *Ontology Encyclopedia of Database Systems*. Springer-Verlag, 2009.
- M. Hammer. Reengineering work: Don't automate, obliterate. *Harvard Business Review*, pages 104–112, 1990.
- M. Hammer and J. A. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Business Books, New York, 1993.
- M. Havey. *Essential Business Process Modeling*. O'Reilly, 2005.
- M. Henkel and J. Zdravkovic. Supporting development and evolution of service-based processes. In *2nd International Conference on e-Business Engineering (ICEBE'05)*, 2005.
- S. Herold, A. Rausch, A. Bosl, J. Ebell, C. Linsmeier, and D. Peters. A seamless modeling approach for service-oriented information systems. In *5th International Conference on Information Technology: New Generations (ITNG'08)*, 2008.
- B. Hicks. Lean information management: Understanding and eliminating waste. *Information Management*, 2007.
- A. Hornung, J. Koschmider, and J. Mendling. Integration of heterogeneous bpm schemas: The case of xpdL and bpel. In *18th International Conference on Advanced Information Systems Engineering (CAiSE'06) Forum*, 2006.
- J. Hu and P. Grefen. Conceptual framework and architecture for service mediating workflow management. *Information and Software Technology*, Vol. 45, Issue 13, 2003.
- C. Huemer, P. Lieg, R. Schuster, H. Werthner, and M. Zapletal. Inter-organizational systems: from business values over bp to deployment. In *2nd IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST'08)*, 2008.
- IEEE. Guide to the software body of knowledge (swebok), 2004.
- ISO. Iso/iec 9126 software engineering – product quality – part 1: Quality model, 2001.
- ISO. Iso/iec 25010 systems and software engineering square (systems and software quality requirements and evaluation) – system and software quality models. <http://www.iso.org/>, 2005-2011.
- S. Jablonski and C. Bussler. Workflow management: Modeling concepts, architecture, and implementation. *International Thomson Computer Press*, 1996.
- I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.
- C. Jimmerson, D. Weber, and D. Sobek. Reducing waste and errors: Piloting lean principles at intermountain healthcare. *Quality & Patient Safety*, 2005.
- F. Jouault and I. Kurtev. Transforming models with atl (atlas transformation language). In *Satellite Events at MoDELS Conference*, 2005.
- N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, 2001.
- B. Karakostas and Y. Zorgios. *Engineering Service Oriented Systems, a model driven approach*. IGI Global, 2008.
- B. Kitchenham. Procedures for performing systematic reviews. Technical report, Software Engineering Group, Department of Computer Science, Keele University and Empirical Software Engineering National ICT Australia Ltd., 2004.
- B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, School of Computer Science and Mathematics, Keele University, Department of Computer Science, University of Durham, EBSE TR, 2007.

- B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El-Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. Technical report, National Research Council Canada, Institute for Information Technology, 2001.
- A. Kleppe, J. Warmer, and W. Bast. *MDA explained: the model driven architecture : practice and promise*. Addison Wesley, 2003.
- N. Kock and F. Lau. Information systems action research: Serving two demanding masters. *Information Technology & People (special issue on Action Research in Information Systems)*, 14 Issue 1:6–11, 2001.
- T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann. Identification and analysis of business and software services- a consolidated approach. *IEEE Transactions on Services Computing*, Vol. 2, No. 1., 2009.
- D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA Best Practices*. Prentice Hall, 2005.
- M. Laguna and J. Marklund. *Business Process Modeling, Simulation and Design*. Prentice Hall, 2005.
- I. Lazarte, Tello-Leal, R. E., J., O. Chiotti, and P. Villareal. Model-driven development methodology for b2b collaborations. In *International Workshop on Models and Model-Driven Methods for Service Engineering (3M4SE'10)*, 2010.
- P. Liew, K. Kontogiannis, and T. Tong. A framework for business model driven development. In *12th International Workshop on Software Technology and Engineering Practice (STEP'04)*, 2004.
- C. Lott and H. Rombach. Repeatable software engineering experiments for comparing defect-detection techniques. *Journal of Empirical Software Engineering*, 1, Issue 3:241–277, 1996.
- L. Maruster and N. van Beest. Redesigning business processes: a methodology based on simulation and process mining techniques. *Knowledge and Information Systems*, 21, Issue 3, 2009.
- R. McTaggart. Principles of participatory action research. *Adult Education Quarterly*, 41 Issue 3, 1991.
- S. Mellor, A. Clark, and T. Futagami. Model driven development. *IEEE Software*, 2003.
- J. Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer, 2008.
- J. Mendling, M. Moser, and G. Neumann. Transformation of yepc business process models to yawl. In *21st. Symposium on Applied Computing (SAC'06)*, 2006.
- J. Mendling, H. Reijers, and W. M. P. van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology*, 52, Issue: 2:127–136, 2010.
- H. Mili, M. Frendi, and et al. Classifying business processes for domain engineering. In *18th International Conference on Tools with Artificial Intelligence (ICTAI'06)*, 2006.
- B. Mora, F. García, F. Ruiz, and M. Piattini. Smml: Software measurement modeling language. In *8th OOPSLA Workshop on Domain-Specific Modeling, (OOPSLA'08)*, 2008.
- B. Mora, F. García, F. Ruiz, and M. Piattini. Graphic versus textual software measurement modelling: An empirical study. *Software Quality Journal*, Vol. 19, Issue 1:201–233, 2011.
- M. Murzek, G. Kramler, and E. Michlmayr. Structural patterns for the transformation of business process models. In *10th International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, 2006.
- M. Myers. Qualitative research in information systems. *MIS Quarterly*, 21 Issue 2:241–242, 1997.
- M. Netjes. *Process Improvement: The creation and Evaluation of Process Alternatives*. Beta, 2010.
- B. Norton. Towards the ontology-based transformation of business process models. In *4th International Workshop on Semantic Business Process Management (SBPM'09)*, 2009.

- OASIS. Service oriented architecture reference model (soa rm). <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>, 2006.
- OASIS. Web services business process execution language (ws-bpel). <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>, 2007.
- OASIS. Service oriented architecture reference architecture (soa ra). <http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf>, 2008.
- OASIS. Ws-bpel extension for people (bpel4people). <http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html>, 2010a.
- OASIS. Web services-human task (ws-humantask). <http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html>, 2010b.
- L. O'Brien, L. Bass, and P. Merson. Quality attributes and soa. Technical report, Software Engineering Institute (SEI), CMU/SEI-20055-TN-014, 2005.
- OMG. Model driven architecture (mda). <http://www.omg.org/cgi-bin/doc?omg/03-06-01>, 2003.
- OMG. Business process definition metamodel (bpdm), versión 1.0. <http://www.omg.org/spec/BPDM/1.0/>, 2008a.
- OMG. Business process maturity model (bpmm). <http://www.omg.org/spec/SoaML/>, 2008b.
- OMG. Query/view/transformations (qvt). <http://www.omg.org/spec/QVT/1.0/>, 2008c.
- OMG. Semantics of business vocabulary and business rules (sbvr). <http://www.omg.org/spec/SBVR/1.0/>, 2008d.
- OMG. Software & systems process engineering metamodel specification (spem). <http://www.omg.org/spec/SPEM/2.0/>, 2008e.
- OMG. Business process modeling notation (bpmn) versión 1.2. <http://www.omg.org/spec/BPMN/1.2>, 2009a.
- OMG. Service oriented architecture modeling language (soaml), beta 2. <http://www.omg.org/spec/SoaML/1.0/Beta2/>, 2009b.
- OMG. Service oriented architecture modeling language (soaml), beta1. <http://www.omg.org/spec/SoaML/1.0/Beta1/>, 2009c.
- OMG. Business motivation model (bmm). <http://www.omg.org/spec/BMM/>, 2010.
- OMG. Business process model and notation (bpmn2). <http://www.omg.org/spec/BPMN/2.0/>, 2011a.
- OMG. Meta object facility (mof). <http://www.omg.org/spec/MOF/>, 2011b.
- OMG. Unified modeling language (uml). <http://www.omg.org/spec/UML/>, 2011c.
- OpenGroup. Service oriented architecture ontology (soa o). <http://www.opengroup.org/projects/soa-ontology>, 2008.
- F. Oquendo. Formal approach for the development of bp in terms of soa using pi-adl. In *4th. IEEE International Symposium on Service-Oriented System Engineering (SOSE'08)*, 2008.
- B. Orriens, J. Yang, and M. Papazoglou. A rule driven approach for developing adaptive service oriented business collaboration. In *3rd International Conference on Services Computing (SCC'06)*, 2006.
- N. Padak and G. Padak. Guidelines for planning action research projects. ohio literacy resource center. <http://archon.educ.kent.edu/Oasis/Pubs/0200-08.html>, 1994.
- M. Papazoglou and W. van den Heuvel. Service-oriented design and development methodology. *Web Engineering and Technology*, Vol. 2, No. 4, 2006.
- M. Papazoglou and W. van den Heuvel. Business process development life cycle methodology. *Communications of ACM*, 50 Issue 10:79–85, 2007.

- 
- M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenge. *IEEE Computer Society*, 2007.
- M. P. Papazoglou and D. Georgakopoulos. Introduction: Service-oriented computing. *Communications of ACM*, 46 Issue 10:24–28, 2003.
- S. Patig and H. Wesenberg. Role of process modeling in software service design. In L. Baresi, C.-H. Chi, and J. Suzuki, editors, *International Conference on Service Oriented Computing (ICSOC-ServiceWave'09)*, volume 5900 of *Lecture Notes in Computer Science*, pages 420–428. Springer Berlin / Heidelberg, 2009.
- S. Pflieger. Experimental design and analysis in software engineering part 1-5. *ACM Sigsoft Software Engineering Notes*, 19, Issue 4:16–20, 1994.
- F. Pino, J. Hurtado, J. Vidal, F. García, and M. Piattini. A process for driving process improvement in vses. In *International Conference on Software Process (ICSP'09)*, 2009.
- M. Poppendieck. Principles of lean thinking. In *OOPSLA (Object-Oriented Programming, Systems, Languages & Applications) Onward!*, 2002.
- D. Quartel, R. Dijkman, and M. van Sinderen. An approach to relate business and application services using isdl. In *9th International Enterprise Computing Conference (EDOC'05)*, 2005.
- T. Rademakers and R. van Liempd. *Activiti in Action: Executable business processes in BPMN 2.0*. Manning Publications and Co., 2011-2012.
- H. Reijers. *Design and Control of Workflow Processes: BPM for Service Industry*. Springer, 2003.
- C. Robson. *Real world research: A resource for social scientists and practitioners-researchers*. Blackwell, 2002.
- E. Rolón, F. Ruiz, F. García, and M. Piattini. Evaluation measures for business process models. In *21st Symposium on Applied Computing (SAC'06)*, 2006.
- S. Roser, B. Bauer, and J. Muller. Model- and architecture-driven development in the context of cross-enterprise business process engineering. In *International Conference on Services Computing (SCC'06)*, 2006.
- A. Rozinat, M. Mans, R.S.and Song, and W. van der Aalst. Discovering colored petri nets from event logs. *Software Tools for Technology Transfer*, 10 (1):57–74, 2008.
- F. Ruiz and J. Hilera. *Ontologies for Software Engineering and Software Technology*, chapter Using Ontologies in Software Engineering and Technology. Springer, 2006.
- P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14 Issue 2:131–164, 2008.
- RUP-IBM. Rational unified process (rup). <http://www-01.ibm.com/software/awdtools/rup/>, 1999-2011.
- N. Russell, A. ter Hofstede, D. Edmond, and W. van der Aalst. Workflow data patterns: Identification, representation and tool support. In *24th International Conference on Conceptual Modeling (ER'05)*, 2005a.
- N. Russell, W. van der Aalst, A. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, 2005b.
- N. Russell, A. ter Hofstede, W. van der Aalst, and N. Mulyar. Workflow control-flow patterns : A revised view. Technical report, BPM Center Report BPM-06-22, 2006a.
- N. Russell, W. van der Aalst, and A. ter Hofstede. Workflow exception patterns. In *18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, 2006b.
- M. Rychly and P. Weiss. Modeling of soa: from business process to service realization. In *3rd International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 08)*, 2008.



- W. Sadiq, S. Sadiq, and K. Schulz. Model driven distribution of collaborative business processes. In *3rd Int. Conference on Services Computing (SCC'06)*, 2006.
- A. Sahai, J. Ouyang, V. Machiraju, and K. Wurster. Specifying and guaranteeing quality of service for web services through real time measurement and adaptive control. Technical report, Management Project, E-Services Software Research Department, HP Laboratories, 1501 Page Mill Road, Palo Alto, CA 94034, 2001.
- D. Schmidt. Model-driven engineering. *IEEE Computer*, 2006.
- R. Shapiro and D. Gagne. What's new in xpdL 2.2. <http://www.wfmc.org/View-document-details/Whats-New-in-XPDL-2.2.html>, 2010.
- A. Sinha and A. Paradkar. Use cases to process specifications in business process modeling notation. In *8th IEEE International Conference on Web Services (ICWS'10)*, 2010.
- H. Smith and P. Fingar. *Business Process Management: The third wave*. Meghan-Kieffer, 2003.
- L. Sánchez González and A. Delgado. Medidas para procesos de negocio y su alineamiento en bpm. In *II Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS'09) en XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD'09)*, 2009.
- L. Sánchez González, A. Delgado, F. Ruiz, F. García, and M. Piattini. *Measurement and Maturity of Business Processes, Handbook of Research on Business Process Modeling*, chapter Measurement and Maturity of Business Processes, pages 532–556. Information Science Reference (IGI Global), 2009.
- L. Sánchez González, F. García, I. Mendling, and F. Ruiz. Assessment and prediction of business process model quality. In *18th International Conference on Cooperative Information Systems (CoopIS'10)*, 2010.
- C. Stahl and M. Volter. *Model-Driven Software Development , Technology, Engineering, Management*. Wiley & Sons, 2006.
- A. Tao Tao and J. Yang. Develop service oriented finance business processes: A case study in capital market. In *3rd International Conference on Services Computing (SCC'06)*, 2006.
- A. Tao Tao and J. Yang. Supporting differentiated services with configurable business processes. In *5th International Conference on Web Services (ICWS'07)*, 2007.
- A. ter Hofstede, W. M. van der Aalst, M. Adams, and N. Russell. Modern business process management: Yawl and its support environment. <http://www.yawlbook.com/home/downloads>, 2009.
- O. Thomas and K. Leyking. Using process models for the design of service-oriented architectures: Methodology and e-commerce case study. In *41st Annual Hawaii International Conference on System Sciences (HICSS'08)*, 2008.
- J. Touzi, F. Benaben, H. Pingaud, and J. Lorré. A model-driven approach for collaborative service-oriented architecture design. *Production Economics*, Vol.121, Is. 1, 2009.
- A. Vallecillo. Model driven development (mdd). <http://www.lcc.uma.es/canal/sabc/MDA-doctorado.pdf>, 2000-2011.
- W. van der Aalst. Workflow course. <http://www.wis.win.tue.nl/wvdaalst/workflowcourse/>, 2002.
- W. van der Aalst and C. Stahl. *Modeling Business Processes-A Petri Net-Oriented Approach*. The MIT Press, 2011.
- W. van der Aalst and M. Voorhoeve. Business process simulation, lecture notes 2ii75, 2010-2011.
- W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14, Issue 3:5 – 51, 2003a.
- W. van der Aalst, A. ter Hofstede, and M. Weske. Business process management: A survey. In *International Conference on Business Process Management (BPM0'3)*, 2003b.

- W. van der Aalst, H. A. Reijers, and A. Medeiros. Business process mining: an industrial application. *Information Systems*, 32, Issue 5:713–732, 2007.
- W. van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell. *Business Process Simulation: How to get it right?*, *Handbook on Business Process Management, International Handbooks on Information Systems*, chapter Business Process Simulation: How to get it right?, pages 317–342. Springer-Verlag, 2010.
- W. M. P. van der Aalst. *Process Mining, Discovery, Conformance and Enhancement of Business Processes*. Number ISBN 978-3-642-19344-6. Springer, 1st. edition, 2011.
- B. van Dongen, A. de Medeiros, H. Verbeek, A. Weijters, and van der Aalst W.M.P. The prom framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets (ICATPN'05)*, 2005.
- W3C. Web services architecture (wsa). <http://www.w3.org/TR/ws-arch/>, 2004.
- Y. Wadsworth. What is participatory action research? action research international, paper 2. <http://www.scu.edu.au/schools/gcm/ar//ari/p-ywadsworth98.html>, 1998.
- I. Weber, J. Hoffmann, J. Mendling, and J. Nitzsche. Towards a methodology for semantic business process modeling and configuration. In *2nd. International Workshop - Business Oriented Aspects Concerning Semantics and Methodologies in SOC (SeMSoC'07)*, 2009.
- M. Weske. *BPM Concepts, Languages, Architectures*. Springer, 2007.
- WfMC. Xml process definition language (xpdl), 2008.
- S. White. Process modeling notation and workflow patterns. Technical report, Business Process Trends, 2004.
- C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*. Springer, 2000.
- M. Wynn, M. Dumas, C. Fidge, A. ter Hofstede, and W. van der Aalst. Business process simulation for operational decision support. In *International Workshop on Business Process Intelligence (BPI'07)*, in *(BPM'07)*, 2007.
- R. Yin. *Case Study Research: Design and Methods*. Sage Publications, Inc, 2002.
- U. Zdun, C. Hentrich, and S. Dustdar. Modeling process-driven and soa using patterns and pattern primitives. *ACM Transactions on the Web*, Vol. 1, No. 3, 2007.
- M. Zelkowitz and R. Wallace. Experimental models for validating technology. *IEEE Computer*, 31 Issue 5, 1998.
- X. Zhao, C. Liu, and Y. Yang. Supporting virtual organisation alliances with relativeworkflows. In *3rd Asia-Pacific Conf. on Conceptual Modelling (APCCM'06)*, 2006.
- M. zur Muehlen. *Workflow-based Process Controlling, Foundation, Design, and Application of Workflow-driven Process IS*. Logos Verlag, 2004.

# Acronyms

BPM	Business Process Management
BPMM	Business Process Maturity Model
BPMN2	Business Process Model and Notation
BPMN	Business Process Modeling Notation
CPI	Continuous Process Improvement
MDA	Model Driven Architecture
MDD	Model Driven Development
MDE	Model Driven Engineering
MOF	Meta-Object Facility
MXML	Mining XML
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
QVT	Query/Views/Transformations Language
SoaML	Service Oriented Architecture Modeling
SOA	Service Oriented Architecture
SOC	Service Oriented Computing
SOD	Service Oriented Development
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WfMC	Workflow Management Coalition
WS-BPEL	Web Services Business Process Execution Language
XPDL	XML Process Definition Language