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

Reporte Técnico RT 13-05

Systematic literatue review

of PSP adaptations

2013

Optimal design o fan IP/NPLS over
DWDM Network.
E. Canale, C. Risso, F. Robledo
ISSN 0797-6410
Reporte Técnico RT 13-03
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 2013

Systematic Literature Review of PSP Adaptations

Silvana Moreno
Álvaro Tasistro
Diego Vallespir

Reporte Técnico InCo/Pedeciba-2013 TR:XXXXX
Junio, 2013

Systematic Literature Review of PSP Adaptations

Resumen Este reporte técnico presenta una revisión sistemática de la literatura existente sobre las adaptaciones al Personal Software Process. En particular, estamos interesados en conocer las adaptaciones al PSP que proponen incorporar el uso de métodos formales. La sección 1 presenta conceptos generales acerca de las revisiones sistemáticas y la sección 2 presenta la revisión sistemática específica realizada.

Palabras claves: Systematic Review, Personal Software Process, Formal Methods

Abstract

In this report we present a systematic review of existing literature about PSP adaptations. In particular, we are interested in getting to know those adaptations that propose to incorporate the use of Formal Methods. Section 1 presents general concepts about systematic reviews. Section 2 presents the specific systematic review carried out.

Contents

1 Systematic Reviews

A systematic review of the literature is a means of identification, evaluation and interpretation of all available information about a research question, subject area or phenomenon of interest (3). Several reasons may justify the realization of a systematic review. It may be necessary to summarize the existing investigations on a certain subject or technology in order to get to know the latter's benefits and limitations. It could also be directed towards identifying shortcomings of current research, so as to suggest further investigations of to provide a frame wherein to position new research activities. Generally, research projects begin with a literature review of some kind. However, this is often not carried out in an exhaustive way, which impoverishes its scientific value. This is the main reason for carrying out systematic reviews. There are normally three phases to a systematic review: planning, conduction and report. We follow guides proposed in (3).

1.1 Planning

During the planning phase the reasons for carrying out the review are identified, the research questions are specified, and the review protocol is defined and evaluated. To state clearly the reasons for performing the review fulfills the important end of confirming its actual need. The research questions determine the goals of the review. It is convenient that these are formulated in terms of one or several questions to be answered during the systematic review. Finally, the protocol of the review specifies the methods to be used during its realization. It is necessary to count on a predefined protocol in order to reduce the possibility of bias on part of the researcher. For instance, in absence of a protocol, the selection of the case studies or their analysis could be influenced by the researcher's expectations. In general, the review protocols are submitted to pair evaluation. The components of a protocol are:

- The reasons for performing the review
- The research questions
- The strategy for the search of articles
- The quality controls
- The strategy for data extraction and synthesis
- The publication strategy

We now clarify the last four points above. The strategy for the search of articles is directed towards generating an adequate search chain and selecting the resources wherein to conduct the search. The search chain is generated by combining in several ways search terms that are derived from the research questions. One general approach consists in decomposing the questions into parts, adding synonyms, abbreviations or alternative expressions for each part, and finally connect the

parts using AND, OR. The resources wherein the search chain is normally applied are digital libraries, as well as repositories of specific journals or conference proceedings. The selection criteria are properties required for an article to be included into the systematic review. The procedures of selection describe how the selection criteria are applied. This is necessary, for instance, to determine how each article is to be evaluated, and disagreements resolved, in the presence of several reviewers. Quality control is normally performed using check lists. These contain the features that the researchers must observe on each article in order to ascertain its quality. In article selection, each feature is ranked, and the sum total of these ranks must overstep a predefined minimal value for the article to be included into the review. The strategy of data extraction determines how to obtain the required information from each study. The strategy of data synthesis determines how the information obtained is to be synthesized. Finally, the publication strategy determines the ways in which the systematic review is to be made generally available, i.e. through journal, brochure, poster, web page, report, etc. Once the protocol is set up, it must be evaluated. The way this is done depends largely on the available budget. Students will generally submit their protocols to their supervisors' judgement. The assessment of the protocol must confirm that the search chain is adequately derived from the research questions and that the procedure of data analysis is suitable for answering such questions.

1.2 Conduction

In the phase of conduction the articles are selected, their quality is secured, and data extraction and synthesis are carried out according to the protocol.

1.3 Report

The final phase of a systematic review consists in writing down their results and making them available generally or to the stakeholders.

2 Systematic Review of Adaptations of the Personal Software Process

In this section we present a systematic review of existing adaptations of the PSP. The division into subsections corresponds to the manner of presenting systematic reviews introduced by Kitchenham (3).

2.1 Reason for the review

The production of software has become a process focused on quality. PSP is a disciplined and individual process directed towards producing quality software. One of its principles consists in finding and correcting software defects at early stages of the development. Our proposal,

PSP_{VDC} is an adaptation of the PSP that incorporates the use of Formal Methods. The objective of this proposal is to improve quality by reducing the number of defects that arrive at the stage of Unit Testing.

The description of the PSP can be found in "PSP: A Self-Improvement Process for Software Engineers" (2). The description of the PSP_{VDC} can be found in "PSP_{VDC}: An Adaptation of the PSP that Incorporates Verified Design by Contract" (4).

The goal of the present review is to know whether there exist other proposals of adaptation of the PSP that aim at improving the quality of the software produced. In particular, we are interested in adaptations that incorporate the use of Formal Methods.

2.2 Research questions

The research questions are the following:

1. Do there exist proposals of adaptations to the PSP?
2. Do they incorporate use of Formal Methods?

2.3 Search strategy

The digital libraries employed in the search have been: SCOPUS, Springer, IEEE Xplore and EBSCO. These search engines comprise the main collections of journals and conference proceedings in the area of Software Engineering. Besides, a manual search was conducted on "A Bibliography of the Personal Software Process (PSP) and the Team Software Process (TSP)", as well as on the collection of proceedings of the TSP Symposium carried out between 2006 and 2012. The first work cited above is a document edited by the SEI that contains references to books, chapters, sections and other types of publications concerned with the PSP and the TSP. In order to encompass a larger number of articles, we also performed a crossed search. That is, we searched for articles that cited any of those articles found by means of our initial search strategy. In this second search we employed the same libraries as in the original search. The search chain employed consists of three parts. The first part is related to the manners of referring to the PSP. The second part considers synonyms of "adaptation", i.e. different manners of referring to changes of the PSP. Finally, the third part refers to "use of Formal Methods". The first part is thus mandatory, whereas the second and third parts are both admissible. Therefore we arrive at the following: (PSP or "personal software process") and ((adapting or extending or over or incorporating) or ("formal methods" or "design by contract")) In applying the search chain on the search engines, we discovered that the acronym PSP is used within a large variety of subjects, which makes the search return very many irrelevant results. Therefore we decided to eliminate the option "PSP", on the basis of the consideration that "personal software process" is likely to appear either in the title or in the abstract of the articles we aimed at. Therefore the search chain finally used is the following:

("personal software process") and ((adapting or extending or over or incorporating) or ("formal methods"' or "design by contract"))

2.4 Criteria of inclusion and exclusion

The author of the present review evaluated each of the articles retrieved by both the automatic and manual search. In evaluating an article, we considered its title, keywords and abstract. Those articles for which at least one of the following conditions is verified are to be excluded:

- The article does not focus on the Personal Software Process.
- It is a book or book chapter.
- It is a duplicate of one coming from a different source.
- It is not written in English.

2.5 Quality Control

We expect to find a very low number of articles satisfying the criteria of inclusion, and so we decide not to perform a quality check on them. Nevertheless, in a hypothetical quality check list we should include:

- That the adaptations proposed are presented in a clear and complete manner.
- That the article is published in a pertinent journal or conference.
- That the article has been cited.

2.6 Data Extraction and Synthesis

The data to be extracted from the selected articles are the following:

- Title
- Kind of publication (journal, magazine, conference, workshop)
- Year of publication
- Abstract
- Results/Conclusions

The data synthesis will consist in classifying the articles on the basis of our research questions, i.e. in: articles adapting PSP by incorporating Formal Methods, articles adapting PSP in any other respect, and articles using Formal Methods together with PSP without making any adaptation of the latter. The results will be studied as related work of the present thesis.

2.7 Results

During the conduct of the systematic review we identified studies. We use the search string on the selected search engines on 07/03/2013. On the IEEE repository, after eliminating contents of types Books & eBooks, 31 results are obtained. On EBSCO results are filtered so as to avoid those contained in CAB Abstracts 1990-Present, Dentistry & Oral Sciences Source, MEDLINE and Ovid Journals. Then 34 results are obtained. Using SCOPUS the items belonging to the areas Life Sciences and Health Sciences are excluded, getting 20 results. Finally, on the Springer repository, by including only Computer Science and English articles, 20 results are obtained. In "A Bibliography of the Personal Software Process (PSP) and the Team Software Process (TSP)" one article is found that proposes to adapt PSP incorporating the use of Formal Methods. This article was also found on EBSCO and IEEE. In the TSP Symposium one article was found that did not appear in the searches conducted on the other sources, and an oral presentation. Eliminating duplicates, the overall results of the primary search are as follows:

- EBSCO 34 articles
- Springer 19 articles
- Scopus 18 articles
- IEEE 29 articles
- TSP Symposium 1 article, 1 oral presentation

After applying the criteria of inclusion and exclusion, there remained 2 articles from the Scopus base, 1 from IEEE, and 1 article and 1 oral presentation from TSP Symposium. Next, the data corresponding to the 4 articles found are presented. They all propose adaptations to the PSP; three of them propose adaptations that incorporate the use of formal methods. We also present the information related to the oral presentation in TSP Symposium that proposes to incorporate to the PSP integration techniques based on models. We got in touch with the authors via email in order to obtain some additional written material on the proposal, but it was not available.

Title: Integrating pair programming into a software development process

Authors: Williams, L.

Type of article: Conference

Year: 2001

Abstract: Anecdotal and statistical evidence indicates that pair programmers - two programmers working side-by-side at one computer, collaborating on the same design, algorithm, code or test - outperform individual programmers. One of the programmers, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects, and also thinks strategically about the direction of the work. On demand, the

two programmers can brainstorm any challenging problem. Because the two programmers periodically switch roles, they work together as equals to develop software. This practice of pair programming can be integrated into any software development process. As an example, this paper describes the changes that were made to the Personal Software Process (PSP) to leverage the power of two programmers working together, thereby formulating the Collaborative Software Process (CSP). The paper also discusses the expected results of incorporating pair programming into a software development process in which traditional, individual programming is currently used (7).

Conclusions: They described the changes made to the PSP to yield the CSP. These changes involved: 1) updating process scripts to document the role of the driver and the observer; 2) adapting data collection forms and analysis reports and 3) altering design and code review procedures. Making these explicit changes to the process cause several implicit, but beneficial, changes to the development environment.

Title: A Combination of a Formal Method and PSP for Improving Software Process: An Initial Report

Authors: Kusakabe, S., Omori, Y. and Araki K.

Type of article: Symposium TSP

Year: 2012

Abstract: Software process is important for producing high-quality software and for its effective and efficient development. The Personal Software Process (PSP) provides a method for learning a concept of personal software process and for realizing an effective and efficient process by measuring, controlling, managing, and improving the way we develop software. PSP also serves as a vehicle to integrate advanced software engineering techniques, including formal methods, into one's own software development process. While formal methods are useful in reducing defects injected into a system, by mathematically describing and reasoning about the system, engineers may have difficulties integrating formal methods into their own software development processes. We propose an approach in which engineers use PSP to introduce formal methods into their software processes. As our initial trial, we followed one of our graduate students as he tried to improve his personal process with this approach. He measured and analyzed his own process data from PSP for Engineers-I, and proposed and experimented with an improved software process with a formal method, the Vienna Development Method (VDM). The experimental results indicate he could effectively reduce defects by using VDM.

Conclusions: Kusakabe, Omori, and Araki proposed an approach in which developers use PSP as a framework of software process improvement to introduce formal methods into their software process for realizing effective and efficient development of high-quality software. They reported one initial trial of the introduction of formal methods into personal process based on PSP. According to the process data in the trial, the developer spent more time in Design and less time in Test. He successfully reduced the number of defects he had focused on without decreasing his productivity.

Title: Adapting the Personal Software Process (PSP) to formal methods.

Authors: Babar, A. , Potter, J.

Type of article: Conference

Year: 2005

Abstract: The goal of good software engineering practice is to deliver reliable, high-quality software on-time and on-budget. In this paper we advocate the combination of two modern approaches towards achieving this goal. On the one hand, with an eye to software quality, we consider adopting a state-based formal development method, the B-method. In terms of tool support and industry adoption, this is the most advanced such method. On the other, aimed at improving the development practices of individual developers, we consider the adoption of the Personal Software Process (PSP). To our knowledge this combination of formal methods and PSP has not been considered before; we term our special version of the combination B-PSP. We present a re-design of the PSP data collection and analysis tasks specifically geared towards the B-Method. Although we support the general framework of PSP, we also believe that developers do not enjoy having their creative and thinking process being interrupted by the need to regularly log activities. With this in mind, we present the PSP tasks in a style which should be acceptable to B developers. We view the results of this paper as a specification for some of the data logging and analysis requirements of a B-PSP-based development (1).

Conclusions: Babar and Potter combine Abrial's B Method with PSP into B-PSP. They add the phases of Specification, Auto Prover, Animation, and Proof. A new set of defect types is added and logs are modified so as to incorporate data extracted from the B machine's structure. The goal of this work is to provide the individual B developers with a paradigm of measurement and evaluation that promotes reflection on the practice of the B method, inculcating the habit of recognizing causes of defects injected so as to be able to avoid these in the future. We have had no notice about further results of this research. In comparison to B, our chosen formal method is significantly lighter and so, we expect, easier to incorporate into actual industrial practice.

Title: VDM over PSP: A Pilot Course for VDM Beginners to Confirm its Suitability for Their Development

Authors: Suzumori, H., Kaiya, H., Kaijiri, K.

Type of article: Conference

Year: 2003

Abstract: Although formal methods seem to be useful, these is no clear way for beginners to know whether the methods are suited for them and for their problem domain, before using the methods in practice. We propose a method to confirm the suitability of a formal method. The method is realized as a pilot course based on the PSP. A course mentioned in this paper is designed for a typical formal method, VDM. Our course also helps beginners of VDM to learn VDM gradually and naturally. During the course, they can confirm its suitability as follows; First, they practice several exercises for software development, while techniques of VDM are introduced gradually. Second, process

data and product data of software development are recorded in each exercise. Third, by evaluating these data by several metrics, they can confirm the suitability of VDM for their work (6).

Conclusions: They propose the combination of VDM and PSP. The Design phase is modified incorporating the formal specification in the VDM-SL language. Besides, the phases of VDM-SL Review, Syntax Check, Type Check and Validation are added. A prototype course is proposed in which each student is to carry out nine exercises applying VDM on the PSP. Data thereby collected shows that about 90% of the defects are eliminated before the Code Review phase. A conclusion is then that the use of VDM contributes to eliminate defect injection during design. After this work was concluded, the research was discontinued for reasons internal to the organization.

Title: Integrating Model-Driven Engineering Techniques in the Personal Software Process

Authors: Pascoal, J.

Type of article: Symposium TSP

Year: 2012

Abstract: The authors propose an approach based on MDE (Model-Driven Engineering) for generating code from models with the objective of checking the quality of the models. The approach consists in developing structural models MDD (for instance, class skeletons) and developing models of partial behavior MBT that are sufficient for generating tests. The PSP is modified to incorporate the construction of the models and the generation of model based tests (5).

Conclusions: The authors conclude that theirs is a "PSP friendly" proposal, which promotes the realization of precise and easily revisable designs at a low cost in terms of script modifications. It is designed to bring short term productivity and quality benefits

Finally, we performed the crossed search on EBSCO, Springer, Scopus e IEEE, for the articles citing any of the 4 found in the original search, which did not add any article to the search results. Using Scopus we found that the articles incorporating the formal method B and VDM into PSP are not mentioned in any other article. Using Springer and EBSCO those two articles are not found. Finally using IEEE we found that the VDM article is quoted in the one using the B formal method in PSP, whereas the latter is not mentioned in any article. The article "A Combination of a Formal Method and PSP for Improving Software Process: An Initial Report" TSP Symposium, is not found by any sources Springer / EBSCO / IEEE / Scopus. Therefore is not possible to know if it was cited. Finally, using Scopus and Springer we found that the article incorporating pair programming to PSP is cited by the articles "Contemporary peer review in action: Lessons from open source development" and "A Repository of Agile Method Fragments" respectively. However these articles do not focus on PSP. Using IEEE we found that the article is cited by other two articles: "A multiple case study on the impact of pair programming on product quality" and "Heterogeneous and homogenous pairs in pair programming: an empirical analysis", which also focus on the PSP. When searching on EBSCO the article that incorporating pair programming to PSP is not found.

2.8 Discussion

We summarize the main conclusions arising from the systematic review. The goal of the review was getting to know the adaptations to the PSP that have been proposed and, in particular, finding out whether any of them proposed to incorporate the use of formal methods. One of the works found proposes to modify the PSP to carry out pair programming in the software development process. The new process is called Collaborative Software Process (CSP). The authors explain how the scripts, templates and forms of the PSP are adjusted to incorporate pair programming. In particular, they describe the modifications that are required to distinguish the tasks corresponding to each role (developer and observer) and the times at which role switch has to be accomplished. Another modification is the use of two design check lists and two code check lists, one for each role. The oral presentation at the TSP Symposium proposes an adaptation to integrating Model-Driven Engineering Techniques in the PSP. The proposed process modifies the design phase for including the development of a design model that describes the external structure of the system and its behavior. This model is to be subsequently refined for describing the internal structure of the system and its behavior. The phase of Design Review incorporates the checking of the model by using a static analysis tool. Finally, the phases of Code and Unit Test incorporate partial generation of code from the models as well as model based test generation. The other articles propose adaptations to the PSP that incorporate the use of Formal methods. Babar and Potter propose a new process called B-PSP that incorporates the B formal method. The proposal by Suzumori, Kaiya y Kaijiri, as well as that by Kusakabe, Omori and Araki, propose the use of VDM within the PSP. All the proposals found modify scripts, templates and forms in order to give support to the new processes. CSP maintains the same development phases of the PSP, incorporating to them the roles of observer and developer. The proposal that integrating model-driven engineering techniques also maintains the same phases of the PSP, modifying several of them to incorporate code and test generation as well as model checking. The proposal VDM over PSP (VDM-PSP) both adds and modifies phases of the development process. This is the same as in our proposal, PSP_{VDC}. VDM-PSP modifies the Design phase to incorporate formal specification in the VDM-SL language. After formal specification, the phase of Design Review of the PSP is carried out. After that, new phases are to be performed: syntax review, syntax check, type check and validation. During syntax review, the user checks the specifications for syntax defects. Besides, a Tool box automatically carries out syntax review, type checking and validation. In this proposal they do not apply proof techniques because VDM-PSP is intended to be carried out by inexperienced students. As different from VDM-PSP, in PSP_{VDC} we decided to add a new phase aimed at formal specification. This makes it possible to obtain information of time spent, defects injected and removed only as a consequence of formal specification. The VDM-SL syntax review phase in VDM-PSP is similar to that of formal specification review in PSP_{VDC}. Both have as goal to

remove the defects injected during the formal specification. The syntax review and type checking phases, as performed automatically by a tool, could be considered similar to the formal specification compile phase of PSP_{VDC}. The validation phase is not clearly presented by the authors. We understand that in this phase the specification is checked with respect to the user's requirements, using a convenient support tool. This phase can be considered similar to the review of the formal specification of PSP_{VDC}. PSP_{VDC} also includes Test case construct and Pseudo code phases which the VDM proposal does not specify. Figure 1 shows the phases of each process. The colors and arrows indicate how PSP_{VDC} phases can be mapped on those of VDM over PSP.

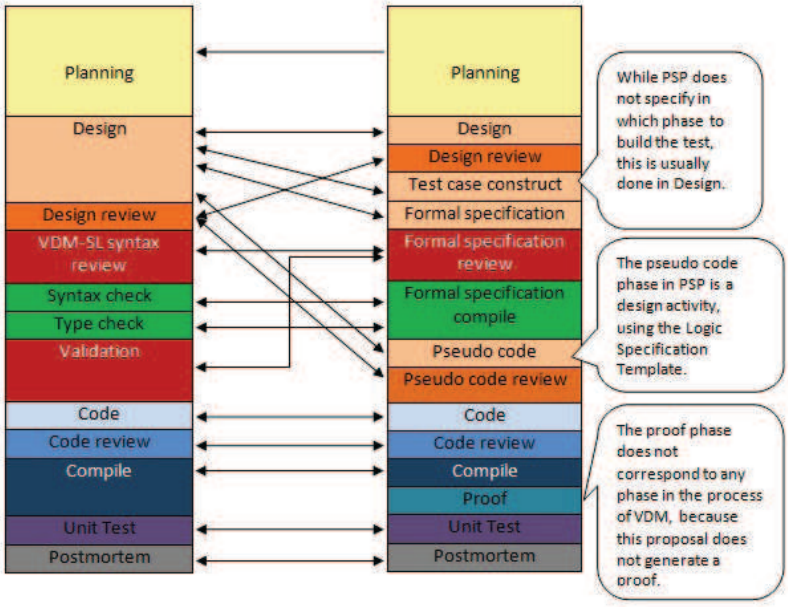

Figure 1: Phases of proposals PSP_{VDC} and VDM over PSP

The proposal by Kusakabe, Omori and Araki incorporates the formal method VDM with its various formal specification languages and a toolkit that enables syntax checking, type checking, the use of an interpreter and the generation of test obligations. The proposal maintains the same phases that PSP, modifying:

- the design phase to incorporate formal specification using any of the specification languages (eg VDM + +) and
- design review phase, to incorporate the use of the VDM toolkit.

As mentioned earlier, this differs from PSP_{VDC}, where we decided to perform the specification and the specification review in new phases. It is not clear in their proposal in which phase the proof obligations are generated. In PSP_{VDC} this is done during the Proof phase. We believe their proposal is incomplete, which makes comparison difficult. We

contacted the authors via email for more information but have got no response. Figure 2 presents the phases of each process. Colors and arrows indicate how PSP_{VDC} phases can be mapped onto those of the proposal by Kusakabe, Omori and Araki.

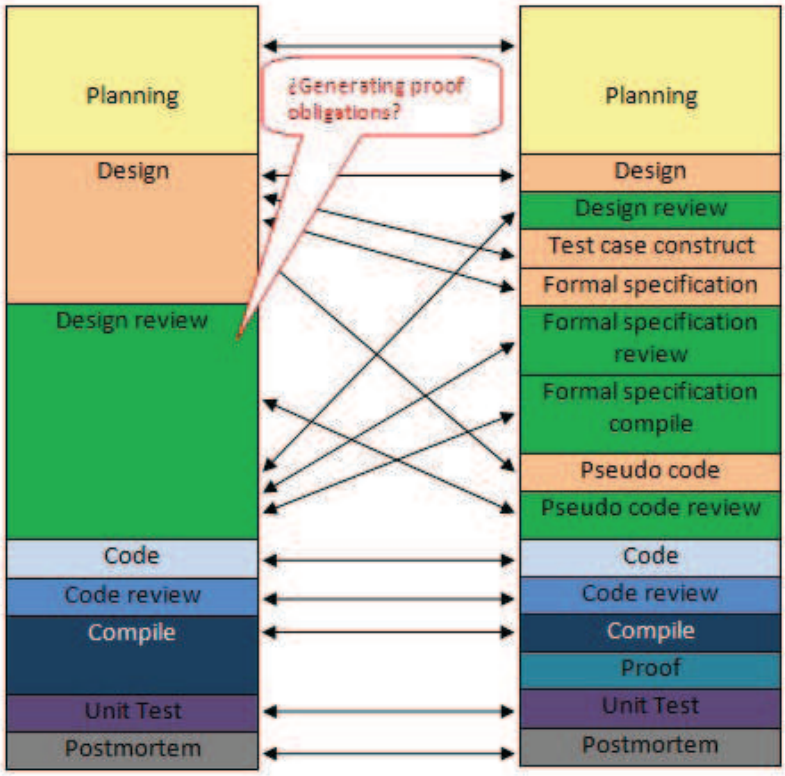

Figure 2: Phases of proposals Kusakabe, Omori and Araki, and PSP_{VDC}

The B-PSP proposal incorporates Specification, Auto Prover, Animation and Auto Proof phases. During the specification the B-machines are specified using the B formal language. Later, Auto Prover and Animation phases are performed with the assistance of the B toolkit. The basic syntax checking and dependency between machines is controlled, generating proof obligations and providing animation. During the Auto Proof an interactive proof of correctness is carried out, using the B toolkit. B-PSP also proposes generating code automatically, but does not explain how that phase is performed. The proposal eliminates the design, design review, code, code review, compile and unit test phases, but does not clarify whether the activities undertaken during these are made in some of the new phases. When comparing B-PSP with PSP_{VDC}, we note that the specification activity is proposed in both cases as a new phase, allowing to collect information about cost and specific defects of that phase. We note that no specification review is performed in B-PSP. PSP_{VDC}, on the other hand, proposes a formal specification review that allows to detect early injected defects. The

syntax control activity performed during the Auto Prover and the Animation phase in B-PSP are carried out in the formal specification review and formal specification compilation phases in PSP_{VDC}. Finally, the Auto Proof phase in B-PSP is analogous to the PSP_{VDC} Proof phase, seeking to generate the proof with the assistance of tools. The figure 3 illustrates the correspondence between the phases of both proposals.

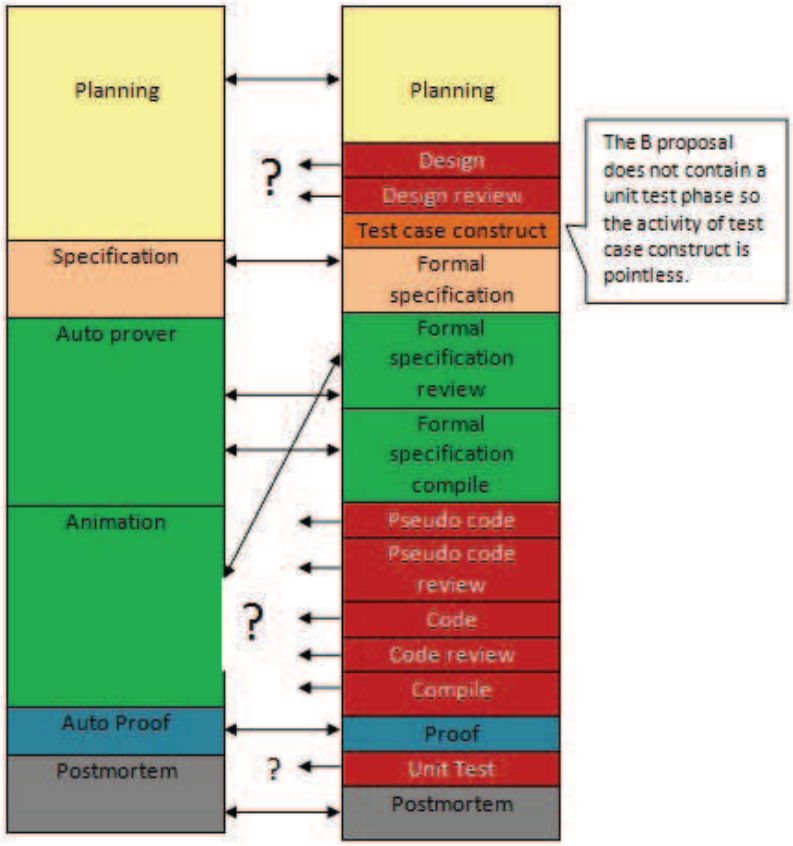

Figure 3: of proposals B-PSP and PSP_{VDC}

3 Conclusiones

We present a systematic review of the literature that attempts to summarize the existing information on the adjustments made to the PSP and particularly those incorporating formal methods. Only 5 articles are found that propose adaptations of the PSP. 3 of them are adaptations of the PSP for the use of formal methods; the other two adapt the PSP to the use of pair programming techniques and integrating model-driven engineering techniques respectively. Unfortunately, the three papers that incorporate formal methods have not presented the full proposed process and do not refer to any technical report doing so to which

we can access. Because of that, it is not possible to make a thorough comparison between these processes and PSP_{VDC}. Nevertheless, we have compared the various phases of the corresponding processes. As detailed in the preceding section, some of the adaptations to the PSP bear similarities to our proposal, while others do not. Different adaptations allow different degrees of granularity of the data collected. The two proposals that incorporate the VDM to the PSP are very different in the way they perform the adaptation. One adds several phases to the process to incorporate the specification and use of tools, while the other has the same phases of the original PSP, incorporating further activity to these. Our proposal PSP_{VDC} is one possible way of adapting PSP by incorporating design by contract. There may be several other proposals for the same purpose. To determine whether PSP_{VDC} produces improvements in individual development process remains now for us to investigate.

Bibliography

(1) A. Babar and J. Potter. Adapting the personal software process (psp) to formal methods. In *Australian Software Engineering Conference (ASWEC'05)*, 2005. 2.7

(2) W. S. Humphrey. *PSP: A Self-Improvement Process for Software Engineers*. Addison-Wesley Professional, 2005. 2.1

(3) B. Kitchenham. Guidelines for performing systematic literature reviews in software engineering. In *EBSE Technical Report EBSE-2007-01*, 2007. 1, 2

(4) S. Moreno, A. Tasistro, D. Vallespir, and W. Nichols. Pspvdc: An adaptation of the psp that incorporates verified design by contract. In *TECHNICAL REPORT CMU/SEI-2013-TR-005 ESC-TR-2013-005*, 2013. 2.1

(5) J. Pascoal. Integrating model-driven engineering techniques in the personal software process. In *TSP Symposium 2012, St. Petersburg, FL*, 2012. 2.7

(6) H. Suzumori, H. Kaiya, and K. Kaijiri. Vdm over psp: A pilot course for vdm beginners to confirm its suitability for their development. In *27th Annual International Computer Software and Applications Conference*, 2003. 2.7

(7) L. Williams. Integrating pair programming into a software development process. In *Software Engineering Education Conference*, 2001. 2.7