Tesis de Maestría
en Ingeniería en Computación

Patrones y Antipatrones de Gestión de Proyectos de Software

Autor: Leonardo Domínguez Pérez

Tutor: MSc Omar Viera

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

Junio, 2009

Dedicado a Maria Fernanda y a Tomás
con todo mi amor

Resumen

El informe de tesis muestra el estudio realizado sobre el estado del arte de los patrones y antipatrones de gestión de proyectos de software. Los antipatrones describen una solución común a un problema, que llevan a consecuencias negativas durante su aplicación, generalmente por no comprender el problema, por no tener el conocimiento o la experiencia suficiente para resolver un determinado tipo de problema en particular, o simplemente por el hecho de haber aplicado un patrón en un escenario que no es el adecuado.

Durante el estudio, se relevan los principales patrones y antipatrones, sus causas y síntomas, sus formas y las consecuencias de sus aplicaciones. Para el caso de los antipatrones estudiamos las soluciones que son refactorizadas y que permiten llegar a un patrón de gestión encontrando soluciones al problema planteado.

Se presenta un caso de estudio y la implementación de un prototipo, en donde mediante técnicas de data mining se analiza la factibilidad de reconocer la aplicación de algunos de los patrones y antipatrones estudiados. Para llevar adelante esta parte del trabajo, se estudian los principales conceptos de data mining aplicables al descubrimiento de patrones. Dado que el área de data mining no es la parte fundamental del trabajo, no se realiza un estudio del arte del tema sino que se presenta una lista de referencias que acompañan el caso de estudio presentado.

Palabras claves: gestión de proyectos de software, ingeniería de software, patrón, antipatrón, data mining.

Abstract

The thesis report shows the survey on the project management software patterns and antipatterns state of the art. AntiPatterns describe a common solution to a problem, leading to negative consequences for their application, usually for not understanding the problem, does not have the knowledge or expertise to solve a certain type of particular problem, or simply because the pattern has been applied in a scenario that is not appropriate for it.

During the study, main patterns and antipatterns causes and symptoms, forms and consequences of their applications are relieved. In the case of AntiPatterns we explore the solutions that are enabling refactoring and that get into a pattern of finding management solutions to the problem.

We present a case study and implementation of a prototype, where with data mining techniques we analyze the feasibility of recognizing the implementation of some of the patterns and antipatterns studied. To carry out this part of the work, we study the main concepts of applying data mining to discover patterns. Since the area of data mining is not the core of the work , a state of the art is not done but we presents a list of references that accompany the proposed case study.

Key words: software project management, software engineering, patterns, antipatterns, data mining.

Contenido Reducido

Contenido Extendido

1 Introducción

"Lo último que se sabe cuando se realiza un trabajo es por dónde empezar."
Pascal
(1623-1662)

La ingeniería de software es un área en constante evolución, que se basa en la generación de conocimiento, la investigación, la experiencia teórica y práctica obtenida de las organizaciones, las comunidades y de las personas que brindan sus aportes a este proceso evolutivo. En esta evolución los diferentes patrones y antipatrones [1.2] que se pueden encontrar en cada una de las ramas de la ingeniería de software, juegan un papel importante al permitir reutilizar estos conocimientos y experiencias, siendo una de las vías que favorecen esta evolución. La gestión de proyectos de software no está ajena a este proceso evolutivo y es una parte fundamental que acompaña cada vez más y fortalece al cumplimiento de sus objetivos.

1.1 Introducción

De forma de introducirnos en el tema podemos comenzar mencionando que un proyecto es un esfuerzo temporal que se lleva cabo para crear un producto, un servicio, o un resultado único, como se define en [PMBOK04], [CLE06], [LEWIS06].

Este esfuerzo temporal es deseable que sea gestionado, con lo cual la gestión de proyectos debe involucrar actividades y tareas que estén relacionadas con el personal, con el producto, con el proceso y con el propio proyecto [PRESS06]. Esta gestión debe lograr un equilibrio que permita cumplir con los objetivos establecidos al principio del proyecto; y que son comunes en todos los proyectos de forma de lograr satisfacer y alcanzar las expectativas que hayan sido generadas.

Los objetivos que se persiguen cuando gestionamos un proyecto están asociados a los propios del proyecto y en ciertos casos también al propio producto o servicio resultante de su ejecución. Es importante tener claros estos objetivos y la diferencia entre ellos, así como también el punto de vista desde donde son observados y medidos.

Los principales objetivos son:

- Cumplir el plan de trabajo y entregar en fecha los productos o servicios resultantes, tal como fueron acordados al inicio del proyecto o en sus diferentes revisiones.

- Ejecutar el proyecto dentro del presupuesto asignado, manteniendo bajo control los costos asociados de forma tal que la rentabilidad del proyecto esté dentro de los márgenes establecidos.

- Mantener la calidad del proceso de gestión, y lograr la misma para el producto o servicio resultante.

- Satisfacer al cliente durante el transcurso del proyecto, con el producto o servicio final.

- Establecer una comunicación y colaboración con el cliente en términos que permitan la generación de futuros proyectos.

- Mantener motivado al equipo participante del proyecto, siendo este uno de los objetivos fundamentales.

Lograr el equilibrio entre todos los objetivos anteriores durante el ciclo de vida de un proyecto de software es el desafío al cual se enfrenta un gerente de proyecto en cada nuevo proyecto de software que comienza. Este equilibrio debe ser gestionado para favorecer el éxito del proyecto, y para el cual una gestión inadecuada puede derivar en su fracaso.

1.1.1 Contexto

Los proyectos existen en cualquier tipo de emprendimiento humano y comprenden diversas actividades y tareas que requieren diferentes habilidades especiales. La diferencia principal entre los proyectos de software y los emprendimientos que son gestionados como organizaciones funcionales clásicas, es que los primeros tienen definido un ciclo de vida; comienzan, se ejecutan y terminan. Un proyecto tiene un número de fases de ciclos de vida, que en forma simple define las fases de concepción, ejecución y finalización [2.1.4].

Dentro del contexto de un proyecto se debe distinguir entre clientes y usuarios. En este informe definimos a los clientes como aquellas personas que están interesadas en el proyecto y que de una manera u otra participan para lograr los objetivos planteados para el proyecto; y como usuarios definimos aquellas personas que hacen uso del producto y/o servicio resultante de la ejecución del proyecto. En el informe se hace referencia a cada uno de ellos según sea el caso.

Para el cliente de un proyecto es necesario que los resultados del proyecto sean integrados, por ejemplo un nuevo sistema de información o un nuevo servicio, dentro de las operaciones del negocio actual de la organización y que los productos o servicios desarrollados cumplan las necesidades para los cuales se los desarrollo. En [FERN99], se enfatiza que el objetivo de la gestión de proyectos en el desarrollo de nuevos productos comerciales no es simplemente la entrega del nuevo producto, si no que es lograr que la organización obtenga un beneficio con este producto dentro del mercado.
En su libro se menciona que la mayoría de los CEOs actuales desean conocer cuando obtendrán los beneficios del producto y también desean tener una previsión de los niveles de beneficios que se generarán, en lugar de saber cuando y a que costo el proyecto estará completo.

Los proyectos tienen diferentes características, existen con diferentes tamaños, son expresados en términos de presupuesto, alcance y duración, y también tienen diferentes grados de riesgo y complejidad. Están los que derivan en productos y los que están asociados a servicios. Dentro de los que pueden ser clasificados según estas características, se encuentran los de ingeniería de software y los de sistemas de información que son los tipos de proyectos que abordamos en este informe. En especial estos tipos de proyectos:

- Requieren diferentes modelos de ciclos de vida.
- Requieren diferentes métodos de control y planificación, sistemas y herramientas.
- Usan terminologías diferentes a los tipos proyectos de otras áreas.
- Demandan diferentes conocimientos, habilidades y experiencia a los gerentes de proyectos, y también a los miembros asignados al equipo del proyecto.
- Poseen distintos énfasis en aspectos de planificación, en estimación de costos, reporte, control, ejecución y en la finalización de un proyecto.

Adicionalmente a las características de una organización, el grado de familiaridad con la tecnología a usar, y la demanda competitiva para iniciar un proyecto son algunos de los factores de ambiente que pueden variar de un proyecto a otro [DESA01].

1.1.2 Gestión de proyectos

La disciplina de la gestión de proyectos ha evolucionado debido a que los principios de la era industrial y los métodos para gestionar a las organizaciones funcionales clásicas no se aplican correctamente en la planificación, el control y la gestión de proyectos, programas y portafolios de proyectos, y en especial a los proyectos de software.

Existen diferentes estudios y teorías relacionadas con las mejores metodologías, estándares y enfoques para realizar una adecuada gestión de proyectos. Se cuenta con una vasta colección de referencias bibliográficas, artículos y papers en donde se indican y presentan diferentes herramientas para gestionar los proyectos. Un análisis y resumen en base a encuestas realizadas a diferentes gerentes de proyectos en relación a las herramientas y sus prestaciones se puede encontrar en [LOCK08].

Las guías y estándares más reconocidos en nuestra región, son provistos por el PMI (Project Management Institute). En 1987, el PMI publica la primera edición del libro *PMBOK® Guide*, que tiene como finalidad identificar las buenas prácticas en la gestión de proyectos y difundir el conocimiento a la comunidad de profesionales del área, guiándolos en la gestión de proyectos [PMBOK04]. El instituto especifica que estas buenas prácticas hacen referencia a la existencia de un acuerdo general en que la correcta aplicación de las habilidades, herramientas y técnicas pueden aumentar el éxito de los proyectos y aclara que "buenas prácticas" no significa que los conocimientos descritos deban aplicarse siempre de forma uniforme en todos los proyectos, sino que son los propios integrantes de los equipos de proyectos los responsables en determinar que es lo más apropiado para aplicar en cada proyecto.

En la guía se definen cinco procesos básicos y nueve áreas de conocimiento comunes a casi todos los proyectos. Los conceptos básicos aplicables a proyectos, programas y operaciones son los siguientes:

1. Inicio
2. Planificación
3. Ejecución
4. Control y Monitoreo
5. Cierre

Los procesos se solapan e interactúan a través de un proyecto, y son descritos en términos de: entradas (documentos, planes, diseños, etc.), herramientas y técnicas (mecanismos aplicados a las entradas) y salidas (documentos, productos, etc.). PMI reconoce la gestión de un proyecto considerando nueve áreas de conocimiento:

1. Gestión de la Integración del Proyecto
2. Gestión del Alcance del Proyecto
3. Gestión del Tiempo del Proyecto
4. Gestión de Costos del Proyecto
5. Gestión de la Calidad del Proyecto
6. Gestión del Riesgo del Proyecto
7. Gestión de Recursos Humanos del Proyecto

8. Gestión de las Comunicaciones del Proyecto
9. Gestión de las Adquisiciones del Proyecto

Para cada una de estas áreas se definen las buenas prácticas con sus correspondientes entradas y salidas. La aplicación de estas prácticas y estándares, puede aumentar el éxito de un proyecto, pero no asegurarlo. En [FOLG05] se menciona que la variable más importante para el éxito de un proyecto son las personas que lo componen y que los verdaderos problemas de los proyectos se encuentran más en el campo de la competencia humana y en su gestión, para lo cual es recomendable contar con un seguimiento de las competencias humanas existentes y necesarias dentro de la organización a través de una herramienta que las gestione en forma adecuada.

El desarrollo de software involucra una mayor cantidad de tareas creativas en relación a otras áreas de negocios. Permitir a las personas que sean creativas e innovadoras, dándoles el ambiente necesario para que puedan desarrollar estas habilidades, es importante, ya que sin ninguna de estas dos características nunca se podrá obtener una ventaja competitiva sustentable, y en un escenario de utilización de nuevas tecnologías, y carente de creatividad e innovación, el equipo no se verá más motivado que a la simple migración de los viejos requerimientos funcionales.

Es necesario controlar y gestionar la comunicación formal e informal de los proyectos usando herramientas de comunicación eficientes [KERZNER03] y es esencial que el diseño y la gestión del proyecto sean simples [CURTIS88]. Lo anterior no significa que simple sea sencillo, ya que por definición la organización de una compañía es tarea complicada. Hay que tener presente la simplicidad en la gestión, tratando de recordar que la dificultad de un proyecto se incrementa exponencialmente con la complejidad de su diseño. [BOEHM88] [BROOKS95] [BROWN96].

1.1.3 Antecedentes

La búsqueda de la simplicidad en la gestión de proyectos motiva el descubrimiento y análisis de patrones y antipatrones de forma que sirva para aprender de los problemas y errores más comunes que ocurren en los proyectos y que partir de estos se tengan en cuenta las soluciones comunes a estas situaciones.

Christopher Alexander, de profesión arquitecto fue quien originó la idea de patrones de diseño, documentando en el año 1977 un lenguaje de patrones para la planificación de ciudades y de la construcción de sus edificios. Alexander cree que la mayoría de los procesos involucrados en el diseño de estructuras físicas es variable, pero que siempre existe un proceso común subyacente a los demás, el cual es el que realmente define los principios de la estructura de diseño y construcción. [ALE77].

El trabajo de Alexander motivó que en el año 1987, Ward Cunningham y Kent Beck desarrollaran un lenguaje de patrones de diseño para el desarrollo de interfaces de usuario dentro del lenguaje Smalltalk [SMALLTALK]. El tema alcanzó un reconocimiento popular en la comunidad de desarrollo de software en el año 1994, cuando Gamma et al., presentan el libro "Design Patterns: Elements of -Reusable Object- Oriented Software" [GOF95], que incluye una serie de construcciones de diseño de software que pueden ser fácilmente aplicadas en la mayoría de los proyectos. A partir de esa fecha han surgido diversas publicaciones, libros, papers y cursos técnicos dedicados a la enseñanza de patrones de diseño. Desde ese entonces se abrió un área de investigación que se dedicó al

estudio de patrones de análisis [FOWLER96] y patrones de arquitectura [FOWLER02], [LARMAN99].

La gestión de proyectos no fue ajena a dicho movimiento, y comenzaron a surgir áreas de investigación en el estudio de patrones y antipatrones de gestión [BMMM98], [BMMM00].

1.2 Planteamiento del problema

El alto índice de proyectos que son cancelados, suspendidos, o que son finalizados con problemas de sobrecostos, problemas de expectativas y calidad del producto, entre otros factores, motiva la investigación y estudio de patrones y antipatrones dentro del área de gestión de proyectos.

Para comprender el problema es necesario estudiar los principales patrones y antipatrones de gestión que describen una solución común a un problema.

1.2.1 Definición de Patrón

Para dar una definición de patrón, nos podemos remitir a [ALE77][ALE79]. Un patrón describe un problema, el cual ocurre en forma repetida una y otra vez en un determinado escenario, y presenta su solución.

> *"Cada patrón es una regla de 3 partes, que expresa una relación entre un contexto, un problema y una solución. Como un elemento en el mundo, cada patrón es una relación entre un contexto, un sistema de fuerzas que ocurren repetidamente en ese contexto y una configuración espacial que permite que esas fuerzas se resuelvan entre sí."*

Alexander en su libro presenta una serie de reglas que son invocadas bajo una determinada circunstancia, que se asemejan a una gramática generativa (1). (Una gramática generativa de un lenguaje intenta dar un conjunto de reglas que predice correctamente que combinaciones de palabras formarán una oración, y que pueden predecir también la morfología de la oración (2) Morfología: campo de la lingüística que estudia la estructura interna de las palabras).

> *"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma."*

1.2.2 Definición de Antipatrón

De la misma forma que existen los patrones, también existen los antipatrones. Este concepto es introducido en [AKROYD96], y tiene foco en la identificación de errores comunes realizados en proyectos de software. Los antipatrones describen una solución común a un problema que lleva a tener consecuencias negativas durante su aplicación, generalmente por no lograr comprender el problema, por no tener el conocimiento y la experiencia suficiente para resolver el tipo de problema en particular, o simplemente por aplicar un patrón en un escenario que no es el adecuado. Los antipatrones examinan las causas, los síntomas y las consecuencias de implementar una solución incorrecta y ofrecen una solución refactorizada, la cual satisface la necesidad requerida por el proyecto de desarrollo de software. [BMMM00].

En [BMMM00] se declara que los antipatrones son métodos que permiten realizar un mapeo eficiente entre una situación general y una clase de solución específica. Los antipatrones proveen experiencia real en el reconocimiento de problemas que son recurrentes en la industria del software y dan una solución para la mayoría de estos problemas. Los antipatrones al igual que los patrones se basan en un vocabulario común para identificar los problemas y discutir sus soluciones en forma efectiva, de forma que sea sencillo comunicar las ideas y soluciones complejas. Los antipatrones dan soporte a la resolución de conflictos, utilizando recursos organizacionales de diferentes niveles donde es necesario, articulando la colaboración entre las distintas fuerzas de diferentes niveles de la gestión y desarrollo del proyecto.

1.2.3 Relaciones entre Patrones y Antipatrones

Los patrones y antipatrones son conceptos relacionados. Lo que distingue a un patrón de otras formas de conocimiento del software es el uso de un template, que sirve para documentar en forma consistente y cubrir adecuadamente la solución a un problema. Generalmente este template justifica la aplicación del patrón y también predice sus consecuencias. En el caso de un antipatrón a diferencia de un patrón, se tienen dos soluciones; la primera es la solución problemática que tiene consecuencias negativas y la segunda la solución refactorizada, que deriva en una mejor forma. Patrones y Antipatrones están relacionados de tal forma que un patrón pueda evolucionar a una solución de antipatrón, evolución que se puede dar en el tiempo y bajo el contexto de su aplicación. Cuando este es el caso, es útil tener alguna forma que permita evolucionar una solución en otra mejor, a través de un proceso de refactoring.

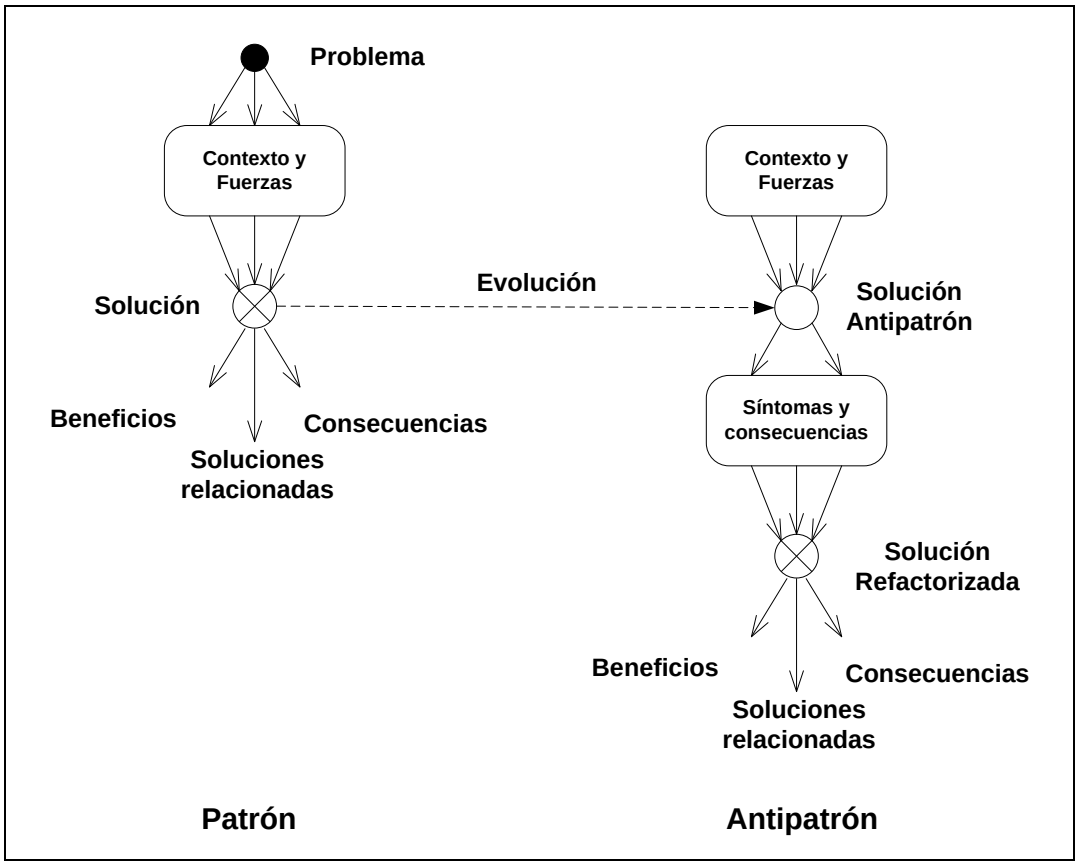


Figura 1 - Concepto de Patrón y Antipatrón

En [BMMM98] Brown et al., presentan antipatrones desde tres puntos vistas diferentes, la del desarrollador, la del arquitecto y la del gerente de proyecto. Los antipatrones de desarrollo describen situaciones encontradas por el programador al momento de resolver problemas de programación. Los antipatrones de arquitectura se concentran en problemas comunes a la hora de definir la estructura de un sistema. Los otros antipatrones se focalizan en describir los problemas y soluciones relacionados con la gestión de proyectos, dentro de una organización. [BMMM00] profundiza específicamente en antipatrones de gestión de proyectos.

El trabajo realizado por Brown, motivó un área de investigación de patrones y antipatrones de gestión de proyectos, que llevaron a la publicación de artículos, papers y libros de diferentes autores, que tienen como modelo de referencia el definido en [BMMM98], [BMMM00]. En [LAP06], Laplante y Neill expanden el alcance de antipatrones de Brown et al., en aspectos relacionados con la gestión y el liderazgo, introduciendo además antipatrones culturales y organizacionales.

1.2.4 Reseña del estado del arte

La gestión de proyectos moderna tuvo su madurez en la época de la segunda guerra mundial simultáneamente en dos industrias: la de construcción e ingeniería y la industria de defensa aeroespacial. Luego de esos años un indicador de su rápida extensión ha sido el patrón de crecimiento que han tenido las asociaciones de gestión de proyectos.

Actualmente las organizaciones reconocen la importancia de realizar una gestión adecuada de proyectos y de llevar adelante la gestión de programas y de un portafolio de proyectos con el fin de satisfacer los objetivos estratégicos que se proponen. Esta gestión está basada principalmente en la utilización de modelos de madurez que proveen procesos y prácticas que guían y ayudan a cumplir con los objetivos del proyecto. Modelos definidos por organizaciones como PMI, APM, PRINCE2 cumplen con objetivos básicos como evaluar las capacidades de gestión de proyectos de una organización, educar y capacitar a las personas involucradas en las tareas de gestión, promoviendo la mejora continua en las habilidades de la organización y de las personas que la componen. Algunos de estos modelos están definidos por el PMI, como es el caso de OPM3, la APM, OGC, y también por la japonesa PMAG. [PMI], [PRINCE2], [PMAG], [OGC].

Estos modelos definen ciclos de vida de proyectos consistentes en un conjunto de fases o etapas, con puntos de evaluación y de decisión, siendo desarrollados y utilizados para soportar las diferentes categorías de proyectos en sus distintas áreas de aplicación.

Existen dos tipos básicos de modelos de ciclos de vida, los modelos predictivos y los modelos ágiles. Los primeros favorecen el concepto de optimización y los segundos el de adaptabilidad.

La planificación de un proyecto es uno de los conceptos que forman parte fundamental de la gestión de proyectos, existiendo la planificación predictiva y la adaptativa las cuales están integradas a las actividades de un proyecto, de forma que todas sus fases y actividades estén lógicamente relacionadas para optimizar el presupuesto asignado y minimizar el tiempo de las actividades incluidas en el plan.

Los métodos, sistemas y herramientas actuales posibilitan a las organizaciones planificar y controlar cada proyecto sobre la base de un ciclo de vida integrado. Incluyen todos los elementos de información como ser, la generación del calendario del proyecto, el manejo de asignación de recursos, el manejo del presupuesto, la información técnica y de riesgo asociada al proyecto, junto con múltiples reportes de gestión. Sistemas que enlazan toda esta información a nivel de un proyecto con el portafolio de proyectos o programas, dando información resumida para los niveles superiores de la organización.

Los paquetes de software actuales integran la información para gestionar un proyecto considerando las funcionalidades más requeridas por los usuarios, dando la posibilidad de intercambiar y administrar esta información entre equipos virtuales o geográficamente distribuidos.

Los proyectos están incrementando en complejidad y no requieren solamente de destrezas de gestión por parte del gerente de proyectos sino que también necesitan de habilidades de liderazgo y principalmente del conocimiento del comportamiento humano.

Lo anterior ha motivado el estudio de patrones y antipatrones de gestión de proyectos donde se destacan principalmente los propuestos por Brown y Laplante. Estos definen un modelo de referencia para su identificación, que tienen su origen en el lenguaje de patrones definido por Alexander. Las categorías que están propuestas actualmente por Brown están relacionadas con las personas, los procesos y la tecnología. Estas categorías son extendidas por Laplante de forma de tener una clasificación de patrones y antipatrones en el ambiente y la cultura de la organización.

Adicionalmente la utilización de metodologías ágiles para el desarrollo de software ha fomentado la elaboración de lenguajes de patrones para prácticas de adopción ágiles y que facilitan la transición de organizaciones hacia un enfoque de gestión de proyectos en forma ágil. Generalmente los patrones ponen más foco en la dinámica de adopción que en los propios derivados de la adopción. [DEAN09],[VAL08].

El crecimiento de las comunicaciones, tecnología y herramientas colaborativas que permiten a diferentes personas interactuar desde diferentes lugares físicos, ha fomentado la gestión de proyectos en forma distribuida y virtual y que a partir de estos aparezcan patrones y antipatrones asociados a la clasificación y ejecución de proyectos distribuidos y virtuales.

1.3 Objetivos

A partir del relevamiento inicial y posterior planteamiento del problema surge el interés y motivación para desarrollar este informe del cual se desprenden los siguientes objetivos:

- *Realizar un estudio del estado del arte en relación a los principales patrones y antipatrones en el área de la gestión de proyectos de software.*

- *Resumir y organizar en una forma fácil de comprender, los principales resultados de la búsqueda e investigación realizada.*

- *Seleccionar alguno de los antipatrones estudiados y plantear un caso de estudio que ayude a la compresión del proceso de su descubrimiento, especialmente estudiando las relaciones a nivel de las comunicaciones existentes dentro de un proyecto.*

El informe no pretende profundizar ni repasar las prácticas formales de la ingeniería de software en relación a la gestión de proyectos, así como tampoco pretende definir una metodología o una guía de gestión, ni tampoco definir estándares, temas que pueden tener un punto de partida en [PMBOK04]. Lo anterior indica que la lectura del informe asume un conocimiento general de temas de gestión de proyectos, particularmente en proyectos de software.

De forma de alcanzar los objetivos anteriores, planteamos un conjunto de objetivos parciales que enumeramos de la siguiente forma:

- *Presentar aspectos relacionados a la situación actual de la gestión de proyectos*
- *Estudiar los modelos de referencia actuales de Patrones y Antipatrones relacionados al área de gestión de proyectos.*
- *Definir un lenguaje de representación de antipatrones de gestión de proyectos en función de los lenguajes estudiados y analizados.*
- *Analizar y diseñar un prototipo para el caso de estudio, basado en datos de proyectos reales.*

La primer parte del informe asociada a cumplir con los dos primeros objetivos mencionados, plantea un análisis descriptivo e interpretativo y no predictivo a través de las diferentes secciones correspondientes al capitulo 2.

La parte del informe relacionada con el caso de estudio se apoya en la utilización de técnicas de data mining para lo cual se estudian los principales conceptos aplicables al descubrimiento de patrones a través de esta técnica. No se realiza un estudio del arte del tema sino que se presenta una lista de referencias que acompaña el caso de estudio.

El caso de estudio presentado está basado en datos de proyectos reales ocurridos dentro de un mismo contexto, en donde los proyectos son tratados en forma uniforme, siendo estos representativos de la realidad planteada.

1.4 Metodología y desarrollo del informe

1.4.1 Metodología

La elaboración de este informe forma parte del proceso de formación académica de la carrera Maestría en Ingeniería en Computación, brindada por el centro de posgrados y actualización profesional del Instituto de Computación (INCO) de la Facultad de Ingeniería de la Universidad de la República. El mismo está comprendido dentro del área temática de gestión de proyectos e ingeniería de software.

La lectura de papers y libros relacionados con las diferentes áreas de interés asociadas al tema propuesto fue el primer paso ejecutado. Este paso forma parte de la metodología realizada para la primera parte de elaboración del informe, la cual tuvo lugar durante el

proceso de investigación y del estudio del arte correspondiente a los patrones y antipatrones. El siguiente paso fue realizar un repaso de los diferentes modelos de madurez y de los ciclos de vida de un proyecto de software, de forma de comprender los contextos en los cuales surgen los patrones y antipatrones de gestión de proyectos, realizando discusiones críticas en cada uno de los puntos.

Una vez realizado los pasos anteriores se planteo la realización de un caso de estudio. Se planteo el problema y se seleccionaron un conjunto de proyectos reales que eran ejecutados en ese momento por la empresa para la cual trabajaba en forma profesional. El caso de estudio fue acompañado por la implementación de un prototipo en donde, mediante técnicas de data mining, se estudia la factibilidad de reconocer la aplicación de algunos de los patrones y antipatrones presentados.

Los pasos anteriores fueron realizados en forma evolutiva, los resultados eran presentados al tutor, el cual daba su feedback, y el mismo era volcado al informe haciendo las correcciones necesarias.

Las presentaciones periódicas de informes y la recolección de feedback formaron parte de la metodología de trabajo y sirvieron para ayudar a mejorar, destrabar y guiar el desarrollo del informe presentado.

1.4.2 Desarrollo

En función de la metodología utilizada para llevar a cabo el trabajo, se enumera a continuación una reseña de los principales momentos surgidos durante el desarrollo de la elaboración del informe:

Agosto 2007: El proyecto de tesis es presentado ante la comisión de posgrados del INCO y aceptado su comienzo.

Setiembre – Octubre 2007: Inicio de recolección de información relacionada a la gestión de proyectos de software, patrones y antipatrones. Lectura de papers, y libros relacionados con el tema.

Noviembre 2007: Presentación del primer borrador del documento del estado de arte, en donde se presentan los conceptos de patrones y antipatrones de gestión de proyectos en base a las definiciones dadas por [ALE77], [BMMM98], [BMMM00], [PMBOK04].

Marzo 2008: Se mejora y presenta la segunda versión del documento del estado del arte, en donde se profundiza sobre los diferentes templates utilizados para la descripción de patrones y antipatrones.

Mayo 2008: Finaliza la tercera revisión del documento del estado del arte, centrado en los modelos de referencia propuestos por Brown, que incluye el estudio de las causas raíces, las fuerzas principales y el modelo de software a nivel de diseño de patrones y antipatrones.

Julio 2008: El documento presentado en su cuarta versión, se focaliza en la clasificación de patrones y antipatrones, teniendo en cuenta la clasificación de Brown, de Laplante, y de los patrones de gestión de proyectos ejecutados en modalidad ágil. También se tiene en cuenta aquellos patrones y antipatrones de proyectos ejecutados en forma distribuida y

virtual. Se realiza un primer análisis de los patrones de comunicaciones en proyectos de ingeniería de software. Se enumeran patrones y antipatrones, y se comienza con el planteamiento de los objetivos para la realización de un caso de estudio.

Setiembre 2008: Se mejora el modelo de referencia, la enumeración y descripción de antipatrones, enriqueciendo la introducción a la gestión de proyectos en base al estado del arte de los modelos de madurez y de los ciclos de vida de los proyectos. Se presenta un relevamiento y las tendencias de uso de las herramientas y sistemas dedicados a la gestión de proyectos por parte la comunidad de gerentes de proyectos.

Noviembre 2008: Se presenta la primera versión del informe, en donde se presenta el resultado del estado de arte junto con el enunciado del caso de estudio, cuyo objetivo es detectar los patrones y antipatrones dentro de las relaciones existentes en las comunicaciones que se dan dentro de un proyecto. Se presenta el problema y el análisis realizado a partir de los e-mails pertenecientes a los proyectos estudiados, que apoyan las soluciones refactorizadas de los antipatrones Irrational Management y MicroManagement.

Febrero – Marzo 2009: Se finaliza el análisis, y diseño de la arquitectura del prototipo. Se implementan los procesos de extracción y transformación de los emails asociados a los proyectos seleccionados para formar parte del caso de estudio.

Abril 2009: Se presenta el primer borrador del informe, incluyendo la introducción, el estado del arte, el caso de estudio y prototipo junto con las conclusiones obtenidas.

Junio 2009: Se realizan mejores a las conclusiones finales y se presenta para revisión.

Julio 2009: Informe final.

1.4.3 Organización

El informe está organizado en diferentes capítulos cuyos contenidos son presentados de la siguiente forma:

Capítulo 1: Se introduce el informe haciendo mención al contexto sobre el cual se desarrolla la tesis, considerando antecedentes y las características de un proyecto y los conceptos de su gestión.

Se presenta la definición del problema que motiva el área de investigación relacionada con los patrones y antipatrones de gestión de proyectos. Se definen los conceptos de patrón, antipatrón y sus relaciones.

Una breve reseña del estado del arte es realizada en donde se plantean los objetivos del informe. Finalmente se comentan los resultados obtenidos junto con sus conclusiones.

Capítulo 2: En este capítulo se muestra el trabajo realizado en relación al estado del arte de la problemática presentada, considerando la situación actual de la gestión de proyectos, los modelos de referencia para la utilización de patrones y antipatrones junto con los templates asociados en cada caso. Se muestra la clasificación de patrones que es usualmente utilizada en la industria y se presenta un modelo para la identificación de estos. Luego se seleccionan y enumeran aquellos patrones y antipatrones que surgen del relevamiento realizado y se detallan los más relevantes.

Capítulo 3: Se presenta el caso de estudio con el objetivo de identificar alguno de los patrones y antipatrones de gestión enumerados en el capítulo anterior.

Capítulo 4: En este último capitulo se presentan las conclusiones y expectativas de trabajos futuros.
Anexos: Sección dedicada a asociada a la información correspondiente a la implementación del prototipo del caso de estudio presentado en el capítulo 3.

Bibliografía: Lista de referencias bibliográficas y links de páginas web mencionadas en las diferentes secciones del informe.

1.5 Conclusiones y Resultados obtenidos

El informe está enfocado al área de patrones y antipatrones de gestión de proyectos de software, en donde la sección correspondiente al estudio del estado del arte incluye como parte central el relevamiento de patrones y antipatrones, sus diferentes clasificaciones, la forma de identificarlos dentro de los contextos y los escenarios usuales en donde se presentan. La enumeración de los patrones y antipatrones de gestión ayuda a tener una mejor compresión de las causas y de las consecuencias que se dan en situaciones generadas dentro del contexto de la ejecución de un proyecto de software. Por este motivo es necesario contar con una forma de describirlos, que permita a los diferentes involucrados una lectura clara. La propuesta del lenguaje presentado en el informe puede ser un buen comienzo para esta descripción.

El estudio del arte realizado en el informe, no arroja un conjunto de herramientas que apoyen al descubrimiento específico de patrones y antipatrones de gestión de proyectos. Es una carencia del área de estudio y la misma debe aportar herramientas y soluciones que ayuden a los gerentes de proyecto o responsables de liderarlos a identificar, prever y recuperarse de las situaciones enumeradas y estudiadas en el conjunto de patrones y antipatrones presentados en el informe. Estos patrones y antipatrones deben ser tenidos en cuenta por las organizaciones de desarrollo de software y por los equipos de proyectos, y ser utilizados como una herramienta más de apoyo a la gestión. Las herramientas deben ayudar a enriquecer el conjunto de patrones y antipatrones, manteniendo actualizado el conjunto en base a las diferentes experiencias y a las lecciones que se aprenden de situaciones reales en la ejecución de proyectos.

El informe aporta los siguientes resultados:

- Forma de describir los patrones y antipatrones utilizando un lenguaje derivado de los propuestos por dos de los autores más referenciados como Brown y Laplante. [2.3]

- Clasificación, identificación y enumeración de las causas, síntomas, y consecuencias de los principales antipatrones de gestión de proyectos. [2.4,2.5,2.6]

- Proceso de formulación de un caso de estudio y método de solución para dar respuestas a un conjunto de preguntas que deben ser formuladas con el objetivo de descubrir algunos de los antipatrones presentados. [3, 3.5]

- Arquitectura para un prototipo propuesto que sirva como punto de partida para el descubrimiento de patrones y antipatrones que ayuden a las organizaciones, gerentes de proyectos y a sus equipos a realizar una mejor gestión de los proyectos en base a datos e informaciones obtenida de experiencias anteriores y actuales. [3.6]

2 Estado del Arte

"Cuánto más se sabe, más se desea aprender. Con el saber crece paralelamente la sensación de no saber o, mejor dicho, de saber que no se sabe."
Sófocles
(496 a. C. – 406 a. C.)

En este capítulo se muestra el trabajo realizado en relación al estado del arte de la problemática presentada en el capitulo anterior. Se considera la situación actual de la gestión de proyectos, los modelos de referencia para la utilización de patrones y antipatrones junto con los templates asociados en cada caso. Se muestra la clasificación de patrones usualmente utilizada en la industria y se presenta un modelo para su identificación. Finalmente se seleccionan y enumeran aquellos patrones y antipatrones que surgen del relevamiento realizado, detallándose los más relevantes.

2.1 Situación actual de la gestión de proyectos

2.1.1 Gestión de portafolio de proyectos y gestión de programas

Los programas se definen como emprendimientos de largo alcance que incluyen dos o más proyectos que requieren de una coordinación cercana entre ellos [ARCH03]. Los proyectos que forman parte de un programa usualmente están relacionados de alguna manera, como ser la utilización de recursos comunes, las relaciones de dependencias entre los proyectos y el soporte a objetivos estratégicos comunes. Por ejemplo, a nivel de tareas que tienen que ser realizadas antes de poder continuar con las tareas de otro proyecto es una muestra clara de dependencia.

Un portafolio es un conjunto de proyectos o programas que se agrupan para facilitar la gestión del trabajo con el fin de cumplir con los objetivos estratégicos del negocio. En la mayoría de las grandes organizaciones se reconoce a los proyectos como otro tipo de inversiones que deben ser gestionadas en base a un portafolio definido.

La gestión de programas es un paso en la dirección correcta, pero la gestión de un portafolio de proyectos de manera más formalizada va más allá de lo que se suele denominar como gestión de programas. En [DYE00] se especifican las diferencias clave existentes entre la gestión de portafolios de proyectos y la gestión de programas. Estas diferencias se muestran en la siguiente tabla:

	Gestión de portafolio de proyectos	Gestión de programas
Propósito	Selección y Prioridad de Proyectos	Asignación de Recursos
Foco	Estratégico	Táctico
Énfasis de la planificación	Mediano Plazo	Corto Plazo

Tabla 1 - Diferencias entre gestión de Portafolios y de Programas.

Un portafolio de proyectos puede consistir de programas y proyectos que soportan una estrategia de alto nivel organizacional.

2.1.2 Modelos de madurez

Los modelos de madurez de gestión de proyectos están centrados en demostrar una mayor capacidad para ayudar y aumentar la probabilidad de lograr los objetivos estratégicos de una organización en forma exitosa a la hora de seleccionar, planificar, ejecutar, controlar y cerrar proyectos y programas [ARCH03].

Los objetivos básicos de estos modelos de madurez son:

- Evaluar las capacidades de gestión de proyectos de una organización.
- Educar y capacitar a las personas involucradas en las tareas de gestión.
- Promover la mejora continua en las habilidades de la organización y de las personas.

Algunos de estos modelos son definidos por diferentes organizaciones como: el PMI [PMI], como es el caso de OPM3 [OPM3], la APM de Inglaterra [APM01], PMMM y

PRINCE2 [PRINCE2] de la OGC oficina de comercio de Inglaterra [OGC], y también por la japonesa PMAG y su modelo P2M [PMAG].

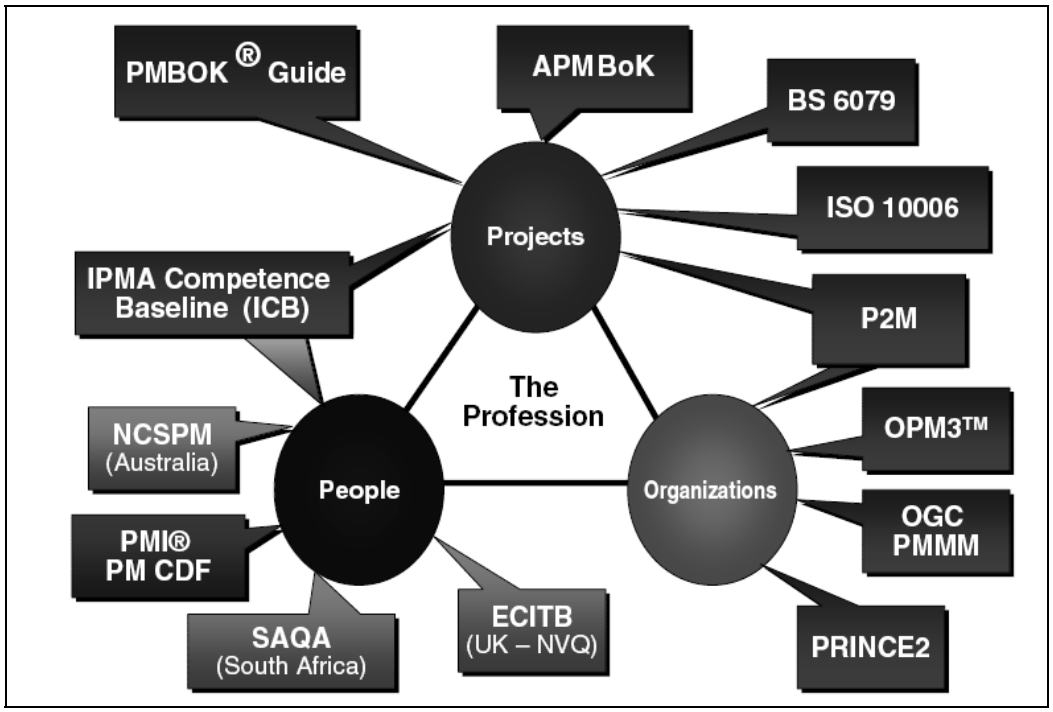


Figura 2 – Estándares con foco en proyectos, organizaciones y personas

Aquellas organizaciones que requieren de un manejo efectivo de proyectos y de sus actividades también manejan el concepto de CPO (Chief Projects Officer), adicionalmente a la utilización de un modelo de madurez determinado. Desde la CPO se tiene una visión global de todos los proyectos que están siendo ejecutados y es de donde parten las directivas estratégicas requeridas por toda la organización. [BIGE03]

2.1.3 Área de aplicaciones de la gestión de proyectos

La gestión de proyectos moderna tuvo su nacimiento durante y en los años siguientes a la segunda guerra mundial, simultáneamente en dos industrias: la industria de la construcción e ingeniería y en la industria de defensa aeroespacial. Luego de estos años un indicador de su rápida extensión ha sido el patrón de crecimiento que han tenido las asociaciones de gestión de proyectos.

Pueden ser identificadas dos tipos de organizaciones, las organizaciones manejadas a través de proyectos y las dependientes de proyectos. En las primeras el negocio principal es construido en base a la implementación de proyectos, las segundas incluyen aquellas organizaciones que tienen como negocio principal el suministro de bienes y servicios, los que no son necesariamente proyectos.

2.1.4 Modelos de ciclos de vida de proyectos

Los modelos de ciclos de vida de proyectos consisten en un conjunto de fases o etapas, que consideran determinados puntos de evaluación y de decisión. Estos modelos son desarrollados y utilizados para soportar las diferentes categorías de proyectos en las distintas áreas de aplicación, y describen los métodos, procedimientos, formularios,

herramientas y sistemas que sirven para realizar la planificación, el análisis y la mitigación de riesgos, presupuesto, monitoreo y control de todo el proyecto.

Según el modelo que se estudie, el nombre de las fases o etapas pueden ser referenciadas en la bibliografía con nombres diferentes. Aún así existe un acuerdo generalizado en que las cuatro fases genéricas de un proyecto son las siguientes [PRESS06]:

- Concepto (iniciación, identificación, selección)
- Definición (factibilidad, desarrollo, demostración, diseño de prototipo, cuantificación)
- Ejecución (implementación, realización, producción, diseño, integración, instalación y test)
- Cierre (finalización, evaluación post mortem, retrospectiva)

Estos modelos requieren que durante cada una de las etapas se identifiquen los productos o artefactos a ser producidos en cada una de ellas. Los puntos clave de evaluación, decisión, eventos o milestones, están definidos para que ocurran al principio y al final de cada fase o subfase del proyecto y en cualquiera de las fases de su ciclo de vida.

Las decisiones autorizan al gerente del proyecto y al equipo a:

- Proceder con el trabajo restante en la fase actual.
- Comenzar el trabajo de la fase siguiente.
- Replanificar y recomenzar una fase o subfase ya completa si los resultados satisfactorios esperados no han sido logrados.
- Revisar las actividades del proyecto y sus planes cuando los cambios de alcance son requeridos.
- Terminar el proyecto si se ha concluido que los objetivos propuestos no pueden ser logrados o que los riesgos que han sido determinados son demasiados grandes.
- Poner el proyecto en suspenso hasta la espera de fondos, tecnología, o de algún evento externo.

Cada proyecto debe documentar y describir su proceso de ciclo de vida, incluyendo cual es el ciclo de vida a utilizar. Se deben especificar los documentos y los niveles de aprobación requeridos para el inicio de nuevos proyectos y cambios de alcance de los proyectos. Adicionalmente se deben identificar los roles clave del proyecto, definiendo sus responsabilidades y niveles de autoridad que tienen sobre el proyecto. Un punto de la documentación es la identificación y descripción de los productos a desarrollar y a entregar en cada una de las fases o subfases del proyecto. Para el tratamiento de los conflictos que se presenten y que no puedan ser resueltos dentro del equipo del proyecto, se debe definir procedimientos para escalarlos, (por ejemplo conflictos que estén relacionados a la competencia de recursos, y a la prioridad entre proyectos)

Existen dos tipos básicos de modelos de ciclos de vida, los modelos predictivos y los ágiles. Los modelos predictivos, favorecen la optimización sobre la adaptabilidad [DESA01]. Entre los principales modelos predictivos encontramos:

- WaterFall [ROYCE70]
- Spiral [BOEHM86]
- Desarrollo Incremental [GILB88]
- RAD (Rapid Application Development) [MARTIN90]
- UP [JACOBSON99]
- Desarrollo basado en Prototipos [PRESS06]

En el año 2001 un conjunto de personas de la industria y de la academia se juntaron para establecer el manifiesto ágil y los principios que lo rigen [AGILE01]. En dicho manifiesto se constan los siguientes valores:

1) Valoramos más a las personas y su interacción que a los procesos y las herramientas.
2) Valoramos más el software en funcionamiento que la documentación exhaustiva.
3) Valoramos más la colaboración con el cliente que la negociación contractual.
4) Valoramos más la respuesta al cambio que el seguimiento de un plan.

En donde explicitan que aunque exista valor en los elementos de la derecha, se valora más los elementos de la izquierda.

Los modelos de procesos ágiles surgen como una reacción a las metodologías predictivas. Estos nuevos modelos buscan un punto intermedio entre no tener ningún proceso y tener demasiados, proporcionando simplemente los procesos necesarios y suficientes, y apostando por la adaptabilidad y flexibilidad a tener en cuenta durante la ejecución de un proyecto de software.

Dentro de los modelos de ciclos de vida ágiles, encontramos aquellos que aceptan en forma natural los cambios surgidos durante el proceso de desarrollo y que en cierta forma se resisten a contar con una planificación detallada al nivel de tareas que deben ser realizadas durante el proyecto, están asociados a los modelos de procesos ágiles y siguen sus valores y principios. Entre estos modelos encontramos:

- SCRUM [SCHW01]
- XP Extreme Programming [BECK99]
- ASD Adaptive Software Development [HSMITH00]
- AUP [AMBLER]

Una vez que el ciclo de vida ha sido definido y documentado para cada categoría de proyectos, entonces es posible definir y documentar el sistema de gestión de ciclo de vida del proyecto. Solamente cuando esta documentación existe, un sistema puede ser mejorado y evolucionado en forma sistemática.

2.1.5 Prácticas, sistemas, herramientas y métodos

No hay una forma única y válida para gestionar cualquier tipo de proyecto. Hay proyectos que tienen como objetivo lograr la eficiencia en el cumplimiento de un plan dentro del presupuesto y tiempos definidos para el producto a desarrollar, mientras existen otros cuyo objetivo es entregar el mayor valor del producto, siguiendo un ciclo de vida ágil en lugar de uno predictivo.

La planificación de un proyecto es uno de los conceptos que forman parte fundamental de la gestión de proyectos. Según lo que vimos, dependiendo del modelo de ciclo de vida, existe la planificación predictiva y la adaptativa que están integradas a las actividades de un proyecto. Integrado significa que todas las fases de un proyecto y todas sus actividades están lógicamente relacionadas. Esta integración es tal que debe lograr optimizar el presupuesto asignado al proyecto, minimizando el tiempo de las actividades incluidas en el plan.

La práctica de planificación predictiva, está basada en que los sistemas que soportan la planificación tienen la capacidad de predecir lo que sucederá en el futuro basándose en los planes y estimaciones que se tengan actualmente del proyecto en curso, considerando continuamente el progreso reportado, actualizando el calendario y el presupuesto del proyecto, y comparándolos con los presupuestos y calendarios; calendarios que definen las líneas base del proyecto generadas al inicio del proyecto y modificadas en los escenarios de cambio, y contra las cuales es posible realizar esta comparación. A través de los conceptos de control y monitoreo del proyecto, para los cuales parte de sus actividades se basan en el resultado de la planificación, se realiza el seguimiento del plan y se intentan predecir los resultados no deseados en forma anticipada, de forma de poder realizar las acciones correctivas necesarias y evitar que los resultados no deseados se conviertan en realidad.

La práctica de planificación adaptativa, tiene como objetivo la entrega de valor de un producto a diferencia de la planificación predictiva en donde veíamos que se busca la eficiencia en el cumplimiento del plan. Esta planificación considera y tiene en cuenta que tanto los requerimientos, como el negocio o el entorno tecnológico puede cambiar durante el desarrollo del producto, con lo cual la planificación debe realizarse considerando iteraciones cortas generalmente no menores a 2 semanas y no mayores a 8 semanas, para lograr al final de la iteración entregar un producto con el mayor valor que satisfaga las necesidades del cliente, y luego adaptar el plan para que considere los cambios que hayan surgido durante el periodo y que den un valor mayor al producto en función de los riesgos y expectativas que el cliente defina en ese momento.

En lo que respecta al aspecto de la gestión de proyectos relacionado con este punto, los métodos, sistemas y herramientas actuales posibilitan a las organizaciones planificar y controlar cada proyecto sobre la base de un ciclo de vida integrado. Incluyen todos los elementos de información como ser, la generación del calendario del proyecto, el manejo de asignación de recursos, el manejo del presupuesto, la información técnica y de riesgo asociada al proyecto, junto con múltiples reportes de gestión. Sistemas que enlazan toda

esta información a nivel de un proyecto con el portafolio de proyectos o programas, dando información resumida para los niveles superiores de la organización.

En [LOCK08] se puede apreciar el resultado de una encuesta que se realizó entre enero y marzo del 2009 con el objetivo de recolectar la experiencia real de los gerentes de proyectos considerando sus experiencias y expectativas de uso en relación con el software de gestión de proyectos que utilizan para gestionar sus proyectos, y analizar la información desde tres puntos de vistas, la profesión de gerente de proyectos, el desarrollo de software de gestión y de la disciplina de investigación de operaciones. La encuesta fue realizada principalmente a gerentes de proyectos. Fueron encuestadas 603 personas de las cuales 501 respondieron a la totalidad de la encuesta. Dentro de los puntos incluidos, se enumeran 27 paquetes diferentes de software de gestión de proyectos, de los cuales los encuestados tuvieron que identificar el paquete de software más utilizado por ellos, el segundo y el tercer paquete en orden de uso.

La lista contenida en la encuesta de paquetes de software sirve como referencia de las principales herramientas de gestión disponibles en la actualidad para llevar adelante la gestión de proyectos.

A continuación se listan los paquetes propietarios y los de código libre.

Sistemas OpenSource	
Aplicaciones de escritorio	Aplicaciones Web
KPlato http://www.koffice.org/kplato/ Open Workbench http://www.openworkbench.org/ TaskJuggler http://www.taskjuggler.org/ GANTT Project http://ganttproject.biz/	dotProject http://www.dotproject.net/ ProjectPier http://www.projectpier.org/ Project.net http://www.project.net/

Sistemas Propietarios	
Aplicaciones de escritorio	Aplicaciones Web
Artemis http://www.aisc.com Cerebral Project(TM) http://www.cerebralproject.com/ Microsoft Project (EPM-Enterprise Project Management) http://office.microsoft.com/en-us/project/default.aspx Merlin http://www.merlin2.net OmniPlan http://www.omnigroup.com Planisware OPX2 Pro http://www.planisware.com/	24SevenOffice http://www.24sevenoffice.com @task http://www.attask.com/ Kiwi Manager http://www.kiwimanager.com/web/home Mingle http://studios.thoughtworks.com/ Project Insight http://www.projectinsight.net/ TaskBin http://www.taskbin.com/ Teamwork

Planview http://www.planisware.com/ Primavera http://www.primavera.com/	http://www.twproject.com/ Wrike http://www.wrike.com/ TOPdesk http://www.topdesk.com

Tabla 2 - Paquetes de software de gestión de Proyectos

Algunos de los resultados de esta encuesta indican que:

- El 48,3 % de los encuestados tiene un grado de Master, y un 3,4% un grado de doctorado, y un 13,9 % de total tiene un grado formal en gestión de proyectos.
- Más de 400 personas encuestadas respondieron tener una certificación PMP® (PMI Project Manager Professional).
- La habilidad de una herramienta de gestión de proyectos para poder planificar las actividades usando métodos CPM/PDM/ PERT y Gantt, es la funcionalidad más importante que debe tener un producto, y le siguen en orden de importancia las siguientes funcionalidades:
    - Capacidad para producir el calendario de tareas maestro en base a la estructura WBS.
    - Brindar el estado del calendario, con la capacidad de predecir en base al progreso realizado y compararlo contra las líneas base definidas.
    - Calculo del camino crítico.
    - Reportes del calendario, incluyendo diagramas de Gantt y de Red.
    - Manejo del presupuesto con control de acuerdo con las líneas base definidas.
    - Asignación de recursos.
- 518 personas respondieron que durante la gestión de sus proyectos utilizaron herramientas para apoyarse en la gestión de sus proyectos. De estas personas 441 respondieron utilizar Microsoft Project como herramienta de gestión de proyectos.

Los paquetes de software mencionados integran la información para gestionar un proyecto considerando las funcionalidades más requeridas por los usuarios, dando la posibilidad de intercambiar y administrar esta información entre equipos geográficamente o virtualmente distribuidos. La importancia de la gestión de proyectos en equipos que

tienen estas características es descrita en [PATT02], quien reconoce esto como la tendencia del mercado.

Otra gama de aplicaciones del tipo ERP, tales como SAP, PeopleSoft, Microsoft Dynamics, OracleSuite, entre otras proveen información acerca de las personas y recursos que pueden ser integrados con las aplicaciones clásicas de gestión de proyectos, de forma de obtener los caminos más efectivos para integrar los proyectos con las operaciones (producción, ventas/marketing, administración, RRHH) en organizaciones que son dependientes de proyectos.

2.1.6 Tendencias

Los proyectos están incrementando en complejidad y no requieren solamente de destrezas de gestión por parte del gerente de proyectos sino que también necesitan de habilidades de liderazgo y principalmente del conocimiento del comportamiento humano. No contar con formación en estas habilidades y basar solamente las decisiones en la información que proveen los paquetes de software, puede ser una de las posibles causas de fallas de los proyectos.

En la gestión de proyectos es más importante comprender el comportamiento humano y las relaciones entre los equipos y los clientes, que contar con una herramienta sofisticada de estimación y planificación. Se debe buscar un balance entre las diferentes áreas de conocimiento de un proyecto, que permita mantener motivado al equipo y lograr que este realice su trabajo en forma efectiva y eficiente.

Es deseable que se pueda ampliar la aplicación de la gestión de proyectos para que la gestión se realice durante todo el ciclo de vida de un proyecto, desde su concepción hasta el momento de generación y obtención de los beneficios del proyecto, para el cual fue pensado, o hasta el propio declive del producto o servicio por el generado.

Las tendencias de gestión de proyectos de software que se perciben son la de continuar con un acercamiento entre la gestión y la estrategia de una organización a través de las prácticas de gestión de portafolios de proyectos, sumadas al descubrimiento de nuevas áreas de aplicaciones para la disciplina de gestión de proyectos, como puede ser la relacionada al estudio de patrones y antipatrones de gestión.

2.1.7 Conclusiones

En esta sección, se presentan algunos aspectos de la situación actual de la gestión de proyectos. Presento en forma breve los conceptos y diferencias entre la gestión de portafolios de proyectos y gestión de programas, según se especifica en [DYE00], de forma de ayudar a entender y comprender los contextos en donde puede ser ejecutado un proyecto determinado, considerando su propósito, foco y el énfasis que se da a su planificación a mediano o largo plazo.

Se presenta el concepto de los modelos de madurez actuales de la gestión de proyectos, mencionando los objetivos básicos. Hago referencia a las organizaciones que actualmente definen estos modelos y que son reconocidos por la industria actual. Todos estos modelos están basados en objetivos comunes, y en donde existe un cierto patrón de asociación de los miembros de las organizaciones que los definen de acuerdo al origen del modelo, por ejemplo la mayor parte de los gerentes de proyecto locales y de la región que está asociado a una de estas comunidades, lo esta al PMI [PMICM], siendo el resto

de las asociaciones no muy difundidas a nivel local y regional. Los modelos de ciclos de vida de proyectos presentados en la sección son los generalmente utilizados en la industria, enumeramos los modelos predictivos y los ágiles, también se muestra un relevamiento de las herramientas más usuales para la gestión de proyectos.

2.2 Patrones y Antipatrones

2.2.1 Modelo de referencia de Patrones y Antipatrones

El modelo de referencia propuesto por Brown en [BMMM00], incluye tres tópicos diferentes dentro de los que se incluyen las *causas raíces*, las *fuerzas principales* y el *modelo de software a nivel de diseño*. A continuación, veremos en detalle cada uno de los puntos que conforman este modelo propuesto.

2.2.2 Causas Raíces

Brown expone en el modelo que las causas raíces son las que principalmente llevan a la falla de los proyectos; fallas que pueden ser del tipo sobre presupuesto, mala planificación y necesidades de negocio no satisfechas, entre otras. Brown enumera las causas raíces en base al comportamiento humano y las identifica como:

- Prisa.
- Apatía.
- Intolerancia.
- Pereza.
- Ambición.
- Ignorancia.
- Orgullo.

Desde el punto de vista de la gestión de proyectos, estas causas raíces pueden ser ejemplificadas como sigue:

- **Prisa:** Las decisiones tomadas con prisa durante las diferentes fases de un proyecto, pueden llevar a afectar a las siguientes y a la larga dar como resultado un producto que no satisfaga las necesidades originales. La generación de estimaciones realizadas en forma apresurada puede llevar a generar propuestas o planes que no son realistas en términos de tiempo y de presupuesto. En este tipo de situaciones se comprometen fases del proyecto, como por ejemplo la de testing. El objetivo pasa a ser la entrega del producto en fecha, sin considerar su calidad. Tomar la decisión de afectar las pruebas unitarias con el objetivo de mantener el presupuesto del proyecto en curso, puede llegar a tener un impacto en la evolución del producto. De la misma forma, la pérdida de tiempo en etapas tempranas, como por ejemplo el tiempo de relevamiento y planificación puede afectar etapas posteriores y críticas como es la etapa de integración, así como también pasar por alto la etapa de diseño de la arquitectura a la larga y en etapas siguientes tiene sus consecuencias.

- **Apatía:** La falta de motivación por parte de la gerencia de proyectos, deriva y se traslada al equipo de proyecto. El trato indiferente y el no ocuparse efectivamente de comprender y resolver los problemas que se presentan, los cuales pueden tener o no una solución conocida terminan afectando al equipo de trabajo y en consecuencia al proyecto.

- **Intolerancia:** Rechazo a la utilización e implementación de soluciones prácticas o a prácticas de gestión efectiva.

- **Pereza:** Esta causa se refleja en las decisiones tomadas por los gerentes de proyectos, en forma sencilla, sin considerar un adecuado análisis de la situación, un análisis de riesgo y que son realizadas con el objetivo de brindar una solución rápida a un problema determinado, pueden llevar a generar problemas en otras dimensiones que puedan afectar al proyecto.

- **Ambición:** La decisión de realizar diseños de arquitectura con detalles excesivos y de la utilización de la última tecnología, o nuevos componentes de software para el equipo del proyecto, así como también la utilización de nuevos métodos de desarrollo puede incrementar la complejidad del proyecto aumentando su riesgo de ejecución. Decisiones de este tipo con exceso de complejidad pueden acarrear problemas de software y cambios en el plan del proyecto que es ejecutado. Los sistemas complejos son caros de desarrollar, integrar, testear, documentar, mantener y evolucionar, lo que lleva a que en muchos casos por causas de la ambición en determinadas etapas, las otras se vean perjudicadas y sean salteadas con el objetivo de recuperar tiempo y presupuesto, lo que puede derivar en fallas del proyecto.

- **Ignorancia:** La falta de conocimiento acerca de técnicas de gestión de proyectos, liderazgo, negociación puede llevar a un gerente de proyectos a tomar decisiones equivocadas que resulten en la falla del proyecto.

- **Orgullo:** El exceso de suficiencia y soberbia por parte del gerente e integrantes del proyecto puede llevar a su falla. La decisión de desarrollar nuevos diseños, y componentes para el proyecto que está siendo ejecutado, en lugar de la reutilización del conocimiento existente en sistemas, productos propios de la organización o de terceros y modelos de desarrollo, cuando lo anterior es posible y que han demostrado ser beneficiosos, hacen de la reinvención un riesgo mayor en tiempo y presupuesto, al tener que ser diseñados, desarrollados y validados antes de que se pruebe su real beneficio al proyecto.

2.2.3 Fuerza Principales

En el modelo de referencia, Brown define las fuerzas principales como los intereses e incidentes que existen en el contexto de la toma de decisiones. Aquellas fuerzas que sean manejadas satisfactoriamente llevan a buenos resultados, las que no, derivan en consecuencias negativas para el proyecto. Algunas de estas fuerzas son de un dominio específico, y reciben el nombre de fuerzas verticales y otras que son aplicables a múltiples dominios, reciben el nombre fuerzas horizontales.

Brown lista las siguientes fuerzas principales:

- **Gestión de las funcionalidades:** Cumplimiento de requerimientos funcionales y no funcionales.
- **Gestión del rendimiento:** Cumplimiento de los requerimientos de rendimiento en la operación del proyecto.
- **Gestión de la complejidad:** Definición del concepto de abstracción.
- **Gestión del cambio:** Control de la evolución del software.
- **Gestión de recursos de IT:** Control del uso de los recursos de IT y de las personas involucradas en el proyecto.
- **Gestión de la transferencia de tecnología:** Control del cambio a nivel de la tecnología.

2.2.4 Modelo de software a nivel de diseño

El modelo de referencia propone como último aspecto un marco de trabajo que permite describir y examinar patrones de acuerdo con una escala. Esta escala presenta diferentes niveles de intereses, desde un nivel granular hasta un nivel global. Los niveles definidos son 7:

- **Objeto:** Es el mínimo nivel granular. Básicamente es el nivel en el cual el desarrollador trabaja con la definición y gestión de clases de objetos y sus instancias. Las decisiones realizadas en este nivel involucran la selección de los objetos y atributos relevantes para cada una de las operaciones a realizar. A nivel del negocio, un objeto puede representar el comportamiento y datos de una entidad de negocio particular. A nivel de desarrollo puede representar los métodos y atributos que lo definen.
- **Micro-arquitectura:** Este nivel incluye diferentes patrones que combinan objetos y clases de objetos que cooperan entre si y cuyas interrelaciones son bien definidas y comprendidas por los desarrolladores del componente. Componente que tiene la responsabilidad de resolver el problema para cual es desarrollado. [GOF95]
- **Framework:** El nivel de framework está relacionado con el desarrollo de patrones de diseño en un nivel macro de componentes, en donde se involucran una o más micro arquitecturas. Los patrones específicos a un determinado framework y aquellos utilizados en un dominio especializado deben ser incluidos en este nivel.
- **Aplicación:** En este nivel se incluyen los patrones de aplicación que cubren un conjunto diverso de soluciones, que involucran diferentes objetos, micro-arquitecturas y frameworks.
- **Sistema:** El sistema comprende uno o varias aplicaciones integradas que juntas proveen las funcionalidades requeridas. En este nivel las fuerzas varían considerablemente respecto al nivel de Aplicación. Se considera que cuanto

mayor sea la escala del sistema, el impacto del cambio y complejidad también aumenta.

- **Empresa:** Es el mayor nivel dentro de la organización, comprende múltiples sistemas con múltiples aplicaciones. Dentro de este nivel se incluyen modelos organizacionales como ser políticas de seguridad y de acceso, comunicaciones y de gestión de recursos. Los patrones de este nivel guían las decisiones de arquitectura que afectan la estructura y la evolución del software.

- **Global/Industrial:** Es el mayor nivel el cual comprende conjuntos de Empresas. El nivel incluye lenguajes, estándares y políticas de software que afectan a las múltiples Empresas.

Las fuerzas definidas en 2.2.3 tienen diferente importancia según el nivel en el que son aplicadas y su grado de impacto puede tener los siguientes valores:

- **Crítico:** El impacto es fundamental, ya que afecta a todo el software.

- **Importante:** El impacto debe ser seriamente considerado, pues afecta a una porción considerable del software.

- **Marginal:** El impacto puede ser a menudo ignorado, ya que afecta solamente a una porción del software.

- **Sin Importancia:** El impacto puede ser no considerado.

	Global/Industria	Empresa	Sistema	Aplicación
Gestión de las funcionalidades	Sin Importancia	Marginal	Importante	Critico
Gestión del rendimiento	Importante	Importante	Critico	Critico
Gestión de la complejidad	Importante	Critico	Importante	Marginal
Gestión del cambio	Sin Importancia	Critico	Critico	Importante
Gestión de recursos de IT	Sin Importancia	Critico	Importante	Marginal
Gestión de la transferencia de tecnología	Critico	Importante	Importante	Marginal

Tabla 3 - Grado de Impacto

En la siguiente tabla se visualizan algunos de los roles que participan del desarrollo de software y su nivel de responsabilidad para cada una de las escalas. Los roles tienen un grado de efectividad dentro del contexto del proyecto y en las diferentes escalas, grado que es identificado en la tabla 3.

	Global/Industria	Empresa	Sistema	Aplicación
CIO	Critico	Critico	Marginal	Sin Importancia
Gerente de Proyecto	Sin Importancia	Critico	Importante	Marginal
Arquitecto	Marginal	Importante	Critico	Importante
Desarrollador	Sin Importancia	Marginal	Importante	Critico

Tabla 4 - Nivel de Responsabilidad

	Global/Industria	Empresa	Sistema	Aplicación
Gestión de las funcionalidades			Arquitecto	Desarrollador
Gestión del rendimiento			Arquitecto	Desarrollador
Gestión de la complejidad		Gerente de Proyecto	Arquitecto	
Gestión del cambio		Gerente de Proyecto	Arquitecto	Desarrollador
Gestión de recursos de IT	CIO	Gerente de Proyecto		
Gestión de la transferencia de tecnología	CIO			

Tabla 5 - Escala de efectividad

2.2.5 Conclusiones

En esta sección se presentaron los modelos de referencias de patrones y antipatrones relevados, particularmente los propuestos por Brown. La bibliografía estudiada en relación con el tema toma como referencias y como base de partida para la elaboración y definición de otros patrones y antipatrones lo descrito por Brown; como es el caso de Laplante y Neill. La enumeración de las causas raíces que Brown propone como las causas que llevan a la falla de proyectos, son aspectos que no se encontraron como parte del relevamiento de las metodologías o modelos de madurez mencionados en el punto 2.1.2. Los modelos mencionados generalmente no consideran la influencia que tiene el comportamiento humano sobre el software, y tampoco las variables culturales de cada organización en donde se vaya a aplicar alguno de estos modelos. Brown incluye acertadamente a las causas raíces basadas en el comportamiento humano y a las fuerzas principales que actúan a la hora de tomar decisiones y Laplante las extiende acertadamente a la organización.

2.3 Templates

Los patrones y antipatrones pueden ser descritos y clasificados para su posterior referencia en base a templates, los que se definen en estructuras básicas y formales. A continuación se presentan algunos de los templates relevados.

2.3.1 Templates de Patrones

Alexander plantea un template para formular su lenguaje de patrones, el cual consta principalmente de 3 secciones.

- **Nombre:** Identifica el concepto.
- **Discusión Técnica:** Contiene la descripción y el contexto del problema.
- ***Therefore*:** Esta palabra tiene el objetivo de separar la discusión entre los problemas y las soluciones que los resuelven.

El patrón anterior es el mínimo necesario para describir cualquier patrón de diseño. Otros de los templates de patrones propuestos en [BMMM98] son:

a) **MicroPattern template:** Es el estándar para patrones de diseño de ingeniería de software. Incluye las secciones: Nombre, Problema y Solución.

b) **MiniPattern template:** En él se descompone el problema o solución en secciones que representan el contexto, las fuerzas, sus beneficios y consecuencias. Estas secciones responden las siguientes preguntas:

   - **Nombre:** Identifica el concepto.
   - **Contexto:** ¿Cuál es el entorno o los supuestos en donde aplicar el patrón?
   - **Fuerzas:** ¿Cuáles son las motivaciones de diseño que hay que balancear?

- **Solución:** ¿Cómo se resuelve el problema?

c) **Inductive Mini-Template, Deductive Mini-Pattern:** el primero de ellos se focaliza en la aplicabilidad del patrón y el segundo en sus beneficios y consecuencias.

Adicionalmente se encuentran los templates formales, que agregan un conjunto de secciones que aportan información apropiada al uso del patrón. Entre estos, se encuentra el propuesto en GoF [GAMA94]. Este template incluye secciones tales como, clasificación de patrones, intención del patrón, variante de nombres, motivación, aplicabilidad, estructura, participantes, colaboración, consecuencias, implementación, código de ejemplo, usos conocidos, patrones relacionados; secciones que hacen a la definición de un patrón en forma formal.

2.3.2 Templates de Antipatrones

Según se menciona en el punto 1.3.2, los antipatrones son nuevas formas de patrones que se diferencian de estos al tener dos soluciones. La primera solución genera consecuencias negativas, en donde existen fuerzas que deben ser solucionadas, y la segunda es el refactoreo de la primera. La segunda solución brinda más beneficios y reduce las consecuencias negativas de la primera.

El template mínimo de un antipatrón incluye las siguientes tres secciones:

- **Nombre:** Identifica el concepto, generalmente en forma peyorativa.
- **Problema Antipatrón:** ¿Cuál es la solución que causa las consecuencias negativas?
- **Solución Refactorizada:** ¿Cómo se evita, minimiza o refactoriza el problema antipatrón?

Muchos libros de gestión de proyectos presentan recomendaciones en formatos que son diferentes de un patrón, por ejemplo [FRAME05] es uno de ellos.

2.3.3 Templates de Antipatrones de Brown

Brown propone un template para los antipatrones que cuenta con secciones requeridas y otras opcionales.

- **Nombre:** Es el nombre que se da al antipatrón, generalmente en forma peyorativa, que lo identifica y que sirve como referencia para los conceptos contenidos en él.
- **También conocido como:** Esta sección identifica nombre y frases que son populares y que también describen al antipatrón.
- **Escala más frecuente:** Indica la escala en la que, según el modelo de software a nivel de diseño definido en 2.2.4, se ubica el antipatrón. Existen situaciones en donde un antipatrón puede ubicarse en más de una escala.
- **Nombre de la solución refactorizada:** Identifica el patrón de la solución refactorizada.

- **Tipo de la solución refactorizada:** Identifica el tipo de solución resultante. La solución puede ser del tipo Software, Tecnología, Proceso, Rol.

    - *Software:* este tipo de patrón generalmente involucra la creación de una nueva solución de software.

    - *Tecnología:* patrones de este tipo, dan solución a problemas a través de la adopción de tecnologías. Puede involucrar desarrollo de software por ejemplo para la adaptación de la tecnología que ha sido adquirida.

    - *Proceso:* provee la definición de las actividades que son consistentemente repetitivas para una solución dada.

    - *Rol:* tipo de solución que implica la asignación de responsabilidades bien definidas para los integrantes de un equipo dentro de la organización y del propio proyecto.

- **Causa Raíz:** Indica cual es la causa raíz que aplica según el modelo definido.

- **Fuerzas desbalanceadas:** Indica las fuerzas primarias, definidas en el modelo, que son ignoradas, mal utilizadas o sobre utilizadas.

- **Evidencia:** Esta sección es opcional. Incluye frases comunes y anecdóticas que hacen referencia al antipatrón.

- **Origen/Historia:** Esta sección es opcional y puede incluir ejemplos de donde se suceden los problemas o información que resulte de interés.

- **Forma General del antipatrón:** Esta sección generalmente provee diagramas y la explicación correspondiente que identifican las características del antipatrón. La solución refactorizada resuelve el antipatrón general expuesto en esta sección.

- **Síntomas y consecuencias:** Enumera la lista de síntomas y consecuencias que resultan de la aplicación del antipatrón.

- **Causas típicas:** Enumera la lista de causas únicas identificadas para este antipatrón adicionales a las descritas anteriormente en la sección de causas raíces.

- **Excepciones conocidas:** Identifica las excepciones al antipatrón, sobre la base de que los comportamientos y procesos del antipatrón no son siempre equivocados.

- **Solución refactorizada:** Explica la solución refactorizada que resuelve las fuerzas del antipatrón identificadas en la sección Forma General. La solución es descrita sin variantes y es estructurada en términos de los pasos de la solución.

- **Variantes:** Es una sección opcional, en donde se listan las variantes más conocidas del antipatrón. Si existen alternativas al mismo, estas son descritas en esta sección.

- **Ejemplos:** Esta sección presenta ejemplos que demuestran como la solución es aplicada al problema.

- **Soluciones relacionadas:** Identifica las citas o referencias cruzadas apropiadas para el antipatrón. Aquellos antipatrones, que estén relacionados con el descrito, y sus diferencias son enumerados en esta sección.

- **Aplicabilidad a otras escalas y roles:** En esta sección se define como el antipatrón impacta en las otras escalas y desde el punto de vista de los restantes roles. También describe su relevancia en los otros niveles.

2.3.4 Templates de Antipatrones de Laplante

En [LAP06], Laplante et al., se define una estructura menos formal a la presentada por Brown, que está enfocada a la identificación de situaciones disfuncionales que se dan en los proyectos.

Las secciones que definen un antipatrón de Laplante son las siguientes:

- **Nombre:** Es el nombre que se da al antipatrón y que expresa el significado del antipatrón. Al igual que en Brown, el nombre es dado en forma peyorativa.

- **Concepto Central:** Esta sección resume al antipatrón, de forma que sea fácilmente identificable.

- **Disfunción:** Indica en términos generales, los problemas con la solución actual, y lista sus posibles síntomas.

- **Vignette:** Ejemplifica el antipatrón en una situación real o hipotética, proporcionando el contexto y el valor del antipatrón.

- **Explicación:** Esta sección amplía la explicación del antipatrón, incluyendo las causas, sus consecuencias y las analogías históricas y/culturales.

- **Band Aid (Curita):** Indica la estrategia o solución a corto plazo, para hacer frente a la situación en forma inmediata, y de esta forma dar tiempo a refactorizar soluciones que requieran de mayor tiempo.

- **Self-Repair:** Sección que ayuda a dar el primer paso en la adopción del antipatrón. Responde a la pregunta de cómo uno mismo se puede ayudar a mejorar, de forma de no caer nuevamente en la situación disfuncional identificada por el antipatrón.

- **Refactoring:** Se describen los cambios que deben ser introducidos para solucionar la situación, junto con su justificación. La solución refactorizada puede necesitar de cambios a nivel de las personas, culturales o de la organización.

- **Observaciones:** Es una sección opcional para realizar comentarios adicionales al antipatrón.

- **Identificación:** Esta sección brinda a través de una serie de preguntas, un instrumento de evaluación informal para el diagnóstico del antipatrón.

2.3.5 Correspondencia entre templates de Brown y Laplante

Si tomamos el template de Brown y de Laplante y consideramos las secciones que pertenecen a la definición de un antipatrón, identificamos que algunas secciones son comunes a ambos y otras pertenecen a uno o a otro.

Si consideramos como *tB* el template definido por Brown y *tL* el definido por Laplante, y *stBi*, *stLj* las secciones correspondientes a *tB* y *tL* con *i,j = 1..n*, es posible encontrar la intersección del conjunto de secciones entre *tB* y *tL*, en donde *stBi* es semánticamente igual a *stLj* para pares *<i,j>*.

$$tB \cap tL = stBi \cap stLj = tBL \text{ con } i,j = 1..n.$$

Dicha intersección da el siguiente conjunto de correspondencia de secciones tBL que definen un template reducido para representación de antipatrones, con las siguientes secciones:

- Nombre = Nombre
- Causa Raíz + Causas típicas + Origen/Historia = Explicación
- Forma General del antipatrón = Concepto Central
- Fuerzas desbalanceadas + Síntomas y consecuencias = Disfunción
- Evidencia + Ejemplos = Vigente
- Excepciones conocidas = Observaciones
- Solución refactorizada + Variantes = Band-Aid + Self-Repair+Refactoring

Las secciones del template de Brown, que no tienen correspondencia con las secciones del template de Laplante son las siguientes:

- **Escala más frecuente:** Está asociado al nivel de diseño definido en el modelo de Brown, que no tiene correspondencia en Laplante.
- **Tipo de la solución refactorizada:** Si existiera la correspondencia con Laplante, esta debería darse en relación con los aspectos sobre los cuales Laplante plantea sus soluciones refactorizadas, es decir, relacionados con cambios a nivel de personas, culturales y de la organización.
- **Soluciones relacionadas:** Laplante no menciona soluciones relacionadas a las propuestas en el antipatrón.
- **Aplicabilidad a otras escalas y roles:** Esta sección esta basada en las escalas definidas por Brown, y no se encuentran una correspondencia clara con las secciones definidas en el template Laplante.

Análogamente la sección del template de Laplante, que no tiene una correspondencia semántica clara con las secciones del template de Brown es:

- **Identificación:** Es una de las carencias que tiene el template de Brown. Aunque Laplante defina esta sección como una evaluación informal, la formulación de un conjunto de preguntas es un buen inicio para comenzar con el reconocimiento y descubrimiento de un antipatrón dentro de una situación determinada y ayuda a la comprensión y aplicación de un antipatrón, ayudando al propio enriquecimiento del lenguaje de patrones.

2.3.6 Conclusiones

Los templates mencionados en esta sección representan los usualmente reconocidos y definen un lenguaje para describirlos. Como se puede notar en 2.3.3 y 2.3.4, los trabajos de Brown y de Laplante presentan diferencias entre las secciones de sus templates de descripción de antipatrones. Como parte del estudio, en el punto 2.3.5 se presentó un template que es el resultado de la identificación de las secciones con significado común a los definidos por Brown y Laplante, de forma que sirvan para posteriores definiciones de antipatrones y que tengan como base un lenguaje común a los definidos por estos autores.

2.4 Clasificación de Patrones

Existen tres aspectos de la gestión de proyectos de software que requieren un control cuidadoso y que están relacionados con: las personas, los procesos y la tecnología. Aspectos que tienen relaciones complejas y que son altamente dinámicos por naturaleza, los cuales deben ser tratados como un todo por los gerentes de proyectos.

Estos aspectos deben ser tratados y balanceados de forma tal de aumentar la probabilidad de éxito de los proyectos. Si cualquiera de ellos es tratado indiferentemente, el proyecto puede llegar sufrir sus consecuencias [MCC95]. La clasificación de patrones y antipatrones relevadas en este informe está centrada en estos aspectos.

2.4.1 Clasificación de Brown

La clasificación de patrones y antipatrones de gestión de proyectos es organizada en [BMMM00], según el aspecto que sea gestionado. De esta forma, se enumera una clasificación básica de Personas, Tecnología y Procesos.

a) Personas

La primera clasificación considera a los patrones y antipatrones, en donde las personas son la causa primaria de los problemas de gestión de proyectos. La interacción entre las personas que participan de un proyecto es crítica para la entrega exitosa del producto-servicio desarrollado en el marco del proyecto. Es necesario que exista una fuerte comunicación y colaboración entre los diferentes roles que participan del proyecto de forma de alcanzar los objetivos que se hayan planteado. Por esta razón en esta primera clasificación se encuentran los patrones y antipatrones que muestran aquellos aspectos de la naturaleza humana que pueden hacer que un proyecto termine bien o mal.

b) Tecnología

Cuando la tecnología es una causa principal de problemas, consideramos los patrones y antipatrones de gestión que caen dentro de esta clasificación. El riesgo en tecnología se ve acrecentado cuando el equipo de proyecto carece del conocimiento de la tecnología utilizada, o es la propia tecnología la que se encuentra en un estado de inmadurez.

En el primero de los casos los roles responsables de este aspecto pueden realizar falsas hipótesis acerca de las posibilidades de la tecnología utilizada en el proyecto, lo cual puede derivar en malas decisiones por parte de gerencia del proyecto, que hagan que el proyecto sufra constantes desviaciones y correcciones del plan de trabajo.

En el segundo caso, la inmadurez de una tecnología, la cual muchas veces es representada por su pobre estabilidad y su débil interoperabilidad entre los diferentes componentes de un sistema, hace que exista una alta probabilidad de ocurrencia de problemas. Ante esto el proyecto debe convivir a través de workarounds y liberaciones continuas de nuevas versiones en pos de lograr llegar a obtener los beneficios que fueron comprometidos a su inicio con la selección de la tecnología en particular.

La tecnología es uno de los aspectos a gestionar y una de las prácticas más débiles de la gestión de los proyectos por parte de los gerentes de proyectos; ya que lo que no puede ser claramente comprendido tampoco puede ser gestionado correctamente.

c) Procesos

En una tercera clasificación encontramos patrones y antipatrones de procesos en la gestión de proyectos. Un proceso, se define como un conjunto de pasos que deben ser seguidos a través de un mecanismo práctico con el objetivo de producir un determinado producto. En [PRESS06], un proceso de software se define como un marco de trabajo para las tareas que se requieran en la construcción de software de alta calidad. Este marco de trabajo establece la base para un proceso de software completo, identificando un conjunto de actividades aplicables a todos los proyectos sin importar su tamaño o complejidad.

Los diferentes procesos pueden tener sus ventajas y desventajas. Un proceso pobre puede ser aquel que está relacionado mayormente con la producción de documentación en lugar de dedicarse a la construcción del producto en sí. La sobrecarga de trabajo de documentación sobre la producción en sí misma, puede hacer que el equipo del proyecto no siga la metodología impuesta en el proceso sobre la creencia que se está siguiendo el proceso equivocado, logrando que la motivación y la productividad del equipo se vean reducidas [DEMARCO99].

Por el otro lado el contar con procesos prácticos bien definidos hace del desarrollo de software una disciplina de ingeniería. Procesos que en determinadas circunstancias deben ser seguidos de forma de generar artefactos necesarios en algunas de las etapas del ciclo de vida del desarrollo de un producto o proyecto. La documentación generada en diferentes etapas del ciclo de vida del un producto o proyecto es beneficiosa para los nuevos integrantes del equipo como también para los integrantes del equipo encargado de mantener y evolucionar un producto. El gerente de proyectos debe conocer y

comprender los beneficios del conjunto de procesos necesarios para el desarrollo de software y utilizarlos en forma práctica.

2.4.2 Clasificación de Laplante

Siguiendo con la clasificación de patrones y antipatrones, [LAP06] introduce un catálogo de antipatrones que complementan los presentados por Brown y extiende la clasificación de patrones y antipatrones al ambiente y a la cultura de la organización en donde se gestionan los proyectos. Según Laplante la clasificación de antipatrones resulta de un conjunto de estrategias corporativas equivocadas y de fuerzas socio-políticas que no son debidamente controladas. Adicionalmente a las anteriores se introduce otro nivel de clasificación basado en una estructura que contempla los siguientes puntos:

- Comunicación
- Competencia
- Coraje
- Cultura
- Finanzas
- Honestidad
- Liderazgo
- Personalidad
- Planificación
- Procesos
- Tecnología

Esta clasificación permite identificar a través de una matriz aquellos antipatrones que afectan la organización. La matriz es expresada horizontalmente por los puntos de clasificación y verticalmente por el conjunto de antipatrones que componen el lenguaje de patrones definido por Laplante. Si se entiende que la organización para la cual se trabaja tiene debilidades en alguno de los puntos enumerados en la clasificación, el usuario puede referenciar la matriz y encontrar cuales son los antipatrones que reflejan su situación.

Laplante sugiere que no se trate de identificar un antipatrón específico, si no por el contrario determinar precisamente cuales son los puntos de la clasificación que son factores influyentes en la situación que se está analizando, y a partir de allí considerar un mayor rango de soluciones a refactorizar.

2.4.3 Patrones de gestión en metodologías ágiles

Desde mediados de los años 90, la creciente popularidad de utilización de metodologías ágiles para el desarrollo de software, tales como XP [BECK99], Scrum [SCHW01], tuvieron su efecto en la gestión de proyectos basados en estas metodologías. Comenzaron a

aparecer lenguajes de patrones para prácticas de adopción ágiles, que incluían patrones de gestión y que ayudaban a las organizaciones a implementarlas. [AEDW06].

En el año 2005, Alistair Cockburn [CBURN05], uno de los impulsores del mundo ágil y firmante del manifiesto ágil, trabajó junto a otro conjunto de personas firmantes del manifiesto en observar cuales de los principios del gerenciamiento podrían ser también requeridos en la gestión de proyectos y de productos en el mundo ágil. Como resultado de dicho trabajo, se publicó en ese mismo año la "declaración de independencia" de la gestión de proyectos ágiles. Esta declaración surgió como el resultado y discusión de las reglas necesarias para obtener un nuevo paradigma de gestión de proyectos modernos, proyectos y gerenciamiento genérico. Los seis principios que ellos sintieron como esenciales para la gestión moderna de proyectos quedaron expresados en esta declaración. Cada uno de los principios está formado según la siguiente cláusula, "Logramos X, haciendo Y", Y es todo aquello que se debe hacer, y la razón por lo cual hay que preocuparse para tratar de obtener X.

Los principios enunciados son:

"Nosotros:

- Incrementamos el retorno de la inversión, haciendo continuo al flujo de valor nuestro foco.
- Entregamos resultados confiables, haciendo participes a los clientes en interacciones frecuentes y en la propiedad compartida.
- Esperamos incertidumbre y la gestionamos a través de iteraciones, anticipaciones y de adaptaciones.
- Fomentamos la creatividad e innovación, reconociendo que las personas son la fuente principal de valor, creando un entorno en donde puedan hacer la diferencia.
- Mejoramos el rendimiento, a través de una responsabilidad grupal por los resultados y una responsabilidad compartida para la efectividad del equipo.
- Mejoramos la efectividad y confiabilidad, a través de estrategias específicas, procesos y prácticas. "

El crecimiento del enfoque ágil en la gestión y desarrollo de software, ha fomentado a un conjunto de organizaciones a intentar adoptar y a adaptarse a este enfoque, en [AMR06], se trata el lenguaje de patrones que facilitan la transición de estas organizaciones hacia un enfoque de gestión de proyectos en forma ágil. Se introducen los patrones que hacen foco en la dinámica de adopción más que en la estructura que resulta de dicha adopción.

2.4.4 Patrones de gestión en proyectos distribuidos

Con el advenimiento de las tecnologías y de las herramientas colaborativas que permiten a diferentes personas interactuar desde diferentes lugares físicos, se comienza a ejecutar proyectos en forma distribuida y virtual.

En [DEEPAK07] se presenta una forma integrada de identificar y aplicar las mejores prácticas para el gerenciamiento de este tipo de proyectos. Presentan una propuesta de

evaluación que permita a los gerentes de proyectos determinar la naturaleza de sus proyectos virtuales y descubrir y aplicar patrones para gerenciarlos. En primera instancia proponen un framework para clasificar a los proyectos virtuales y en segunda instancia ilustran como un gerente de proyectos puede utilizar la propuesta para identificar, describir y usar los patrones para una efectiva gestión del proyecto.

El framework está basado en un conjunto de conceptos relacionados con las características de proyecto virtual, virtualidad y tecnología. En [PP06] se detallan estas definiciones y se realiza un estudio con profundidad en los proyectos virtuales usando la teoría de patrones.

Un proyecto virtual puede ser definido como una alianza de un conjunto de personas que se encuentran en forma dispersa y que trabajan en forma conjunta para realizar actividades y tareas específicas bajo restricciones de recursos y tiempo [SHEN98] [ZIGURS07].

El concepto de Virtualidad, es aquel en cual los miembros de un proyecto están dispersos en una o más ubicaciones y que confían en la tecnología de la información para llevar a cabo los objetivos del proyecto. La confianza en la tecnología es un componente fundamental de la virtualidad, en donde la dispersión no está asociada solamente a la geografía, el tiempo o afiliación organizacional. [DUBE04] [PINS05] [POWELL04]

En [DEEPAK07] se proponen tres dimensiones fundamentales para clasificar a los proyectos distribuidos:

- Complejidad: refleja las características que deben ser gestionadas para la finalización exitosa del proyecto. Características tales como el tamaño del equipo, cultura, lenguaje, género, características personales, recursos, innovación y conocimiento.

- Alcance: El alcance del proyecto define la duración del proyecto y las funcionalidades que debe tener el producto final.

- Riesgo: abarca amenazas y oportunidad que pueden negativamente o positivamente afectar la finalización exitosa del proyecto.

La tecnología para proyectos virtuales es definida como un conjunto de herramientas integradas y flexibles para el soporte de proceso, tareas de análisis y rendimiento y para la comunicación entre los integrantes del proyecto, que según [DEEPAK07] [NUNA91] [ZIGURS98], son dimensiones críticas de la tecnología.

La comunicación puede ser definida como el proceso por el cual las personas transmiten un significado de una a otra a través del cual intercambian mensajes e información en orden a través de algún medio de forma de llevar a cabo las actividades del proyecto [KHA05].

La coordinación es definida como el mecanismo a través del cual las personas y los recursos de tecnología son combinados para llevar adelante las actividades especificas de forma de lograr los objetivos establecidos. [CROW91] [GRANT96]. Control se define como el proceso de monitorear y medir las actividades del proyecto de forma de anticipar

y gestionar los desvíos del plan del proyecto y de los objetivos de la organización, [PMBOK04] [HEND92] [KIRSCH96].

[DEEPAK07] utiliza cada una de las tres dimensiones de las características de un proyecto más la dimensión virtualidad y proponen tres tipos de proyectos: Lean, Extreme e Híbridos [KHA05].

- Proyectos Lean: Tienen baja complejidad, alcance reducido y riesgo relativamente bajo. Estos proyectos tienden a ser fácilmente subdivididos en partes administrables debido a la claridad de las salidas y requerimientos del proyecto.

- Proyectos Extreme: Tiene una alta complejidad, un alcance amplio, y un riesgo alto. Estos proyectos generalmente son de misión crítica y requieren de una actividad intensa y de participación activa de un número de equipos y clientes.

- Proyectos Híbridos: Tienen niveles variados de las tres dimensiones de complejidad, alcance y riesgo.

2.4.5 Conclusiones

Las secciones 2.4.1 y 2.4.2 presentan una clasificación de los patrones y antipatrones, según la propuesta de Brown que está basada en las personas, los procesos y la tecnología, y de acuerdo con Laplante quien hace una extensión de los anteriores considerando el ambiente y la cultura de la organización en donde se gestiona un proyecto de software. La clasificación anterior puede ser enriquecida con los patrones de gestión en metodologías ágiles y de gestión de proyectos distribuidos, en donde se generan otro tipo de patrones y antipatrones, que pueden ser estudiados en [CBURN05],[DEAN07], [DEAN09], [APAT], [DEEPAK07], [VAL08].

2.5 Identificación de Patrones

En [DEEPAK07] se menciona que los patrones para los proyectos virtuales pueden ser identificados y los patrones efectivos pueden ser distinguidos de los no efectivos [KHA05].

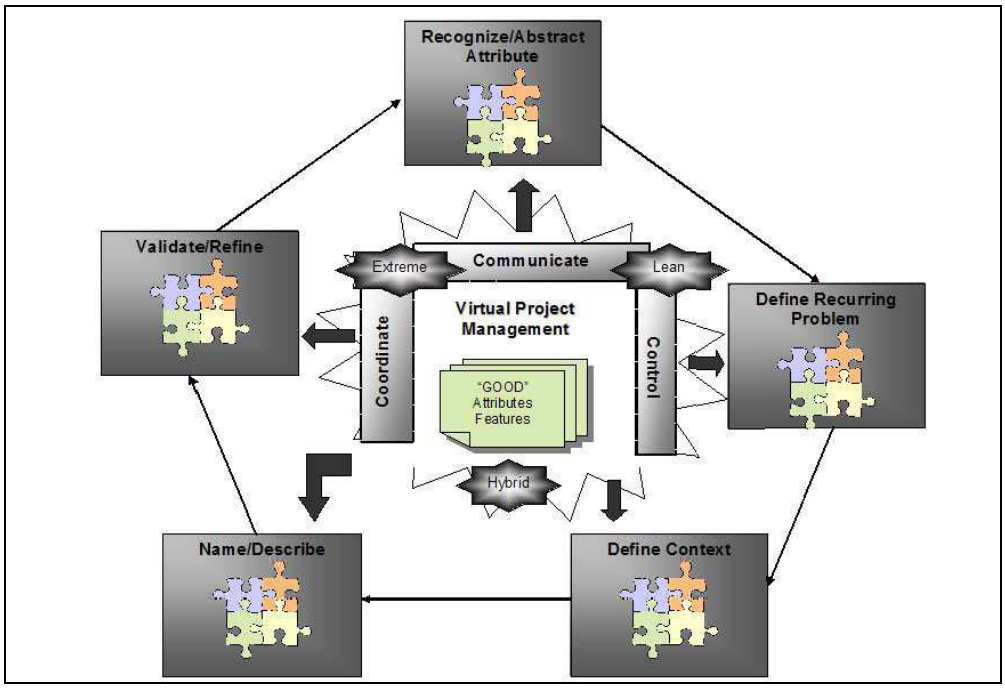


Figura 3 - Framework para identificación de patrones en proyectos distribuidos

[DEEPAK07] argumenta que un patrón de diseño para la gestión de proyectos virtuales debe incluir una descripción de procesos, mejores prácticas, factores, herramientas y técnicas que repercuten sobre la coordinación, comunicación y control. Cada patrón es especifico al tipo de proyecto (lean, extreme, hibrido) y es descrito en términos del patrón de Alexander (nombre, contexto, problema, solución).

2.5.1 Condiciones necesarias

Según [DEEPAK07] para que los gerentes de proyectos puedan extraer patrones para la gestión de proyectos virtuales en forma efectiva, comunicarlos y continuar su evolución, se deben tener las siguientes condiciones:

- Un vocabulario común y una compresión de los conceptos que guían los principios que gobiernan el desarrollo de patrones.

- Un lenguaje simple, natural que permitan tanto a los gerentes experimentados y al principiante la posibilidad de aportar patrones sin dificultad.

- Los patrones deben ser fácilmente de leer, de comprender, de compartir y de comunicar.

- Es deseable la existencia de un repositorio central de patrones en donde compartir el conocimiento dentro de la organización.

- Un modelo en donde cualquier miembro del equipo pueda identificar potenciales patrones, los cuales debieran pasar por la validación de un gerente con mayor experiencia en el área de interés.

- Las personas que han sido exitosas en la gestión de proyectos tienen el conocimiento y habilidades para ayudar, guiar y enseñar a los gerentes más jóvenes.

Los elementos arriba mencionados enumeran el conjunto mínimo de prerrequisitos que deben ser tenidos en cuenta con el objetivo de construir una librería de patrones que sean usables. Una vez que estas condiciones están establecidas, el gerente puede comenzar a extraer y escribir patrones.

2.5.2 Pasos para el descubrimiento de patrones

En [DEEPAK07] se plantean un conjunto de pasos necesarios para llevar adelante el proceso de descubrimiento de patrones.

- Reconocer y abstraer un atributo/característica de la gestión de un proyecto virtual que afecte la comunicación, la coordinación y el control.

- Definir el problema recurrente.

- Definir el contexto en el cual la característica es apropiada.

- Nombrar y describir el patrón.

- Validar y refinar el patrón.

El descubrimiento de patrones puede surgir de sesiones de brainstorming y de la utilización de técnicas de data mining apropiadas.

En el primero de los casos, se pueden presentar ejemplos de preguntas a ser utilizadas para identificar y examinar atributos o facetas de los proyectos, que se perciban como potenciales patrones. Se tomaron ejemplos de [KHA05] en donde se realizó un virtual focus group, conducido a través de herramientas tecnológicas de colaboración.

Ejemplo de extracción de patrones a través de sesiones de brainstorming.
Preguntas

- ¿Qué prácticas específicas implementadas por el equipo del proyecto y la gestión del mismo, contribuyen a la efectividad de su proyecto? Trate de pensar abiertamente en función de comportamientos, tecnologías y herramientas.
    - Por ejemplo, un miembro del equipo podría mantener reuniones regulares para monitorear el avance del proyecto.

- ¿Qué prácticas especificas implementadas por el equipo del proyecto y a su gestión, contribuyen a la no efectividad del mismo? Trate de pensar abiertamente en función de comportamientos, tecnologías y herramientas.

- Por ejemplo, un miembro del equipo podría mantener información importante del proyecto oculta en lugar de compartirla con el grupo.

2.6 Enumeración de Patrones y Antipatrones

A continuación se lista un conjunto de patrones y antipatrones de gestión que desde el punto de vista descrito por Brown y mencionados en 1.3.3, tengan en cuenta solamente al gerente de proyecto.

Según las diferentes formas de describir un patrón y un antipatrón presentados en la sección 2.3, se optó por el template de Brown, al ser este el más completo en términos de las secciones definidas para cada una de las alternativas estudiadas. Se considera el siguiente subconjunto de secciones:

- **Nombre**
- **Escala mas frecuente**
- **Nombre de la solución refactorizada**
- **Tipo de la solución refactorizada**
- **Causa Raíz**
- **Fuerzas desbalanceadas**
- **Síntomas y consecuencias**
- **Causas típicas**
- **Solución refactorizada**

2.6.1 Analysis Paralysis

Nombre: Analysis Parálisis.
Escala mas frecuente: Sistema.
Nombre de la solución refactorizada: Iterative-Incremental Development.
Tipo de la solución refactorizada: Software.
Causa Raíz: Orgullo, Intolerancia.
Fuerzas desbalanceadas: Gestión de la complejidad.

Síntomas y consecuencias:

- Existen múltiples reinicios del proyecto, mucho trabajo a nivel del modelo, debido a cambios en el personal y en la visión del proyecto.
- Los incidentes a nivel de diseño e implementación son continuamente reintroducidos en la etapa de análisis.
- El costo del análisis excede las expectativas sin un punto de predicción.
- La etapa de análisis no involucra la interacción con el usuario. Mucho del análisis realizado es especulativo.
- La complejidad del análisis resulta en implementaciones no sencillas, que hacen al sistema difícil de desarrollar, documentar y testear, y posteriormente en mantenerlo.
- Las decisiones de diseño e implementación son realizadas a nivel de la etapa de análisis.

Causas típicas:

- El proceso de gestión asume un modelo del ciclo de desarrollo en cascada.

- La gerencia tiene más confianza en su habilidad para analizar y descomponer el problema que en el propio diseño e implementación.
- La gerencia insiste en completar todo el análisis antes de que la fase de diseño comience.
- Los objetivos de la fase de análisis no están bien definidos.
- La gerencia no esta dispuesta a tomar decisiones acerca de las partes del dominio que no estén suficientemente descritas.
- La visión del proyecto y el entregable al cliente son difusos.

Solución refactorizada:
El proceso desarrollo iterativo incremental es la solución refactorizada. El proceso de desarrollo en cascada, asume que toda la información y el conocimiento del problema deben ser conocidos de antemano, en cambio, en un desarrollo iterativo incremental, se asume que el problema puede tratarse en iteraciones, en donde en cada una de ellas se puede pasar por un proceso de análisis, diseño, codificación, verificación y validación, construyendo de esta forma el producto en forma incremental. El uso de metodologías ágiles afiliadas al manifiesto ágil, pueden ser una solución a este problema.

2.6.2 Death by planning

Nombre: Death by planning.
Escala mas frecuente: Empresa.
Nombre de la solución refactorizada: Rational Planning.
Tipo de la solución refactorizada: Proceso
Causa Raíz: Avaricia, Ignorancia, Prisa.
Fuerzas desbalanceadas: Gestión de la complejidad.

Síntomas y consecuencias:
Los síntomas usualmente incluyen alguno de los siguientes:

- Incapacidad para planificar a un nivel práctico y simple.
- Foco en el costo más que en la propia entrega del producto.
- Mayor gasto en tiempo de planificación, detalle de progreso, replanificación, en lugar de centrarse en la entrega del producto.
  - El gerente de proyecto planifica todas las actividades del proyecto.
  - Los líderes de equipo, planifican las actividades del equipo, y las de los propios desarrolladores.
  - Los desarrolladores particionan todas sus actividades en tareas.

Las consecuencias incluyen:

- Ignorancia del estado del desarrollo del proyecto. Planes sin sentido que no son actualizados, perdida del control de entrega del producto a medida que el proyecto transcurre.
- Imposibilidad de cumplir con entregas críticas.
- Las consecuencias anteriores crecen incrementalmente llegando a situaciones tales que:
  - Es necesario realizar más inversiones.
  - El proyecto entra en crisis.
  - Se produce la cancelación del proyecto.
  - Es posible perder parte del staff del proyecto.

- Cada planificador tiene que monitorear y capturar el progreso al nivel mostrado por su plan y volver a estimar.
- Planificación sin fin y replanificación causa mayor planificación y replanificación.
- El objetivo pasa a ser la entrega de planes en lugar de la entrega del producto. La gerencia asume lo anterior porque como se realiza el seguimiento del esfuerzo y el costo, el progreso en el producto a entregar es equivalente, siendo esta correlación no directa.
- Desvíos continuos en la entrega de software, y fallas eventuales del proyecto.

Causas típicas:
Las causas típicas están basadas en la carencia de un enfoque práctico, con sentido común para planificar y controlar el progreso del desarrollo del producto.

- Existencia de un plan sin actualizar que muestra los componentes del producto a entregar con fechas de entregas desactualizadas.
- Ignorancia de los principios básicos de gestión de proyectos.
- Planificación inicial demasiado optimista.
- Existencia de un plan que es utilizado como herramienta para lograr la venta del proyecto.
- Planificación como la actividad primaria del proyecto.
- Conformidad forzosa a nivel del cliente.
- Conformidad forzosa a nivel ejecutivo.

Solución refactorizada:
La solución al problema se basa en la elaboración de planes que estén centrados en forma primaria en los entregables del producto, sin tener en cuenta cuantos equipos-personas trabajan en el proyecto con el fin de cumplir con el objetivo delineado. Los entregables pueden ser categorizados en dos niveles, el primero centrado en el propio producto, es decir, aquellos artefactos que fueron comprometidos al cliente y el segundo en los componentes que forman parte del producto.
Los entregables pueden incluir:

- Documento de especificación de requerimientos.
- Documento de arquitectura.
- Criterios de aceptación.
- Escenarios con casos de uso del producto.

El plan debiera incluir hitos en donde realizar la validación de cada uno de los componentes, tales como:

- Aprobación del diseño conceptual
- Aprobación del diseño de especificación
- Aprobación del desarrollo
- Aprobación del plan de test.

El plan debe ser actualizado semanalmente de forma asegurar el control y la reducción de riesgos. El seguimiento es realizado de acuerdo con el nivel estimado de completitud. Los diagramas Gantt pueden ser utilizados efectivamente para mostrar visualmente los entregables, los hitos, fechas y dependencias. La comparación contra la línea base del proyecto debe arrojar por ejemplo los siguientes resultados para los entregables del producto: en tiempo y presupuesto, y completado.

Es fundamental contar con una línea base inicial y que la misma no cambie frecuentemente, lo cual haría perder la habilidad de hacer el seguimiento. Al momento

de realizar la estimación del proyecto es aconsejable considerar un periodo de contingencia para poder tratar con los imprevistos tales como:

- Cambios en requerimientos
- Workarounds en componentes de terceros
- Identificación de defectos
- Correcciones de defectos

Se debiera establecer un tiempo mínimo de ejecución de tareas, de forma de asegurar su ejecución.

En el caso de las metodologías ágiles, se puede realizar la implementación del framework Scrum y utilizar sus buenas prácticas. Tener iteraciones cortas de 30 días (sprints), con objetivos bien definidos y un plan de entregables estimado en forma inicial en días, reuniones de seguimientos a través de reuniones diarias de 15 minutos, en donde cada uno de los integrantes del equipo responda a preguntas tales como:

- ¿Qué tareas se realizaron desde la reunión anterior?
- ¿Qué tareas se tienen para realizar hasta la próxima reunión?
- ¿Existe alguna traba que me impida continuar con alguna tarea en particular?

Esta reunión tiene como objetivo sincronizar al equipo y medir cuanto falta para llegar a los objetivos pautados al inicio del sprint.

2.6.3 Corncob

Nombre: CornCob.
Escala mas frecuente: Empresa.
Nombre de la solución refactorizada: CornCob removal service.
Tipo de la solución refactorizada: Rol.
Causa Raíz: Avaricia, Orgullo, Ignorancia.
Fuerzas desbalanceadas: Gestión de recursos, Gestión de transferencia de tecnología.

Síntomas y consecuencias:
Los síntomas usualmente incluyen alguno de los siguientes:

- Un equipo de desarrollo o un proyecto no progresan debido a la existencia de una persona que está en desacuerdo y es reacia con los objetivos claves o con los procesos definidos a seguir durante el proyecto, y continuamente trata de cambiarlos.
- Una persona esta continuamente elevando objeciones al proyecto en forma de quejas acerca del rendimiento, la confiabilidad, el producto, el cliente y otros.
- El comportamiento destructivo es conocido por muchas personas a nivel de la organización, pero es tolerado y soportado por la gerencia de diversas formas, ya que desconocen o no quieren reconocer el daño causado.
- Las fuerzas políticas crean un ambiente en donde es dificultoso mantener las discusiones técnicas por el buen camino.
- Las fuerzas políticas resultan en cambios frecuentes al alcance o requerimientos del sistema. El proyecto se vuelve mucho más reactivo que proactivo, porque todo el mundo responde a las interminables mejoras propuestas por el CornCob.

- A menudo, la persona destructiva es el gerente el cual no esta bajo la autoridad directa de un gerente senior o de niveles superiores.
- La organización no ha definido un proceso de toma de decisiones que posibilite la resolución de incidentes y que permita actuar en función de estas decisiones. Esto permite al gerente interferir inapropiadamente por fuera de su área de responsabilidad.

Causas típicas:

- La gerencia soporta el comportamiento destructivo, desconociendo el impacto de las acciones que lleva adelante la persona que tiene este comportamiento. El punto de vista de la gerencia es la provista por la propia persona.
- La persona tiene una agenda oculta, que entra en conflicto con la del resto del equipo.
- Existe desacuerdos y fallas de comunicación entre el equipo de trabajo.
- La gerencia no aísla al grupo, de fuerzas internas y externas, y asigna roles en forma inapropiada, que abusan de estos para sus propios fines. La gerencia falla en la asignación de compromiso.

Solución refactorizada:

La solución al problema es aplicada en diferentes niveles de autoridad, incluyendo estrategia, operación y táctica. En todos los casos la acción clave pasa por eliminar el soporte a los comportamientos destructivos por parte de la gerencia en los niveles mencionados.

- Soluciones Tácticas: son soluciones utilizadas on the fly, por ejemplo en una reunión, e incluyen
  - Transferencia de responsabilidad: Dar a la persona responsabilidad total para resolver el problema identificado con un acuerdo de tiempo de solución. En otras palabras, pasar la responsabilidad para planificar y resolver el problema a la persona que eleva las quejas.
  - Aislar el incidente: A menudo, la persona tiene una opinión aislada. Apelando a los intereses del grupo es una forma efectiva para defender contra un comportamiento individual o quejas aisladas. Si la persona es confrontativa, se debe recordar no tomar ninguna cosa en forma personal, hacer esto es siempre un error. Facilitar al grupo la discusión, encuadrando las claves objetadas y tratar de ganar consenso anulando la confrontación. Una simple encuesta es suficiente para demostrar los intereses del grupo.
  - Cuestionar las preguntas: Cuando la persona utiliza palabras o frases ambiguas o sobrecargadas de significado, solicitar la clarificación de las mismas. Cuando se utilice palabras o frases que surjan de rumores, solicitar la justificación de estos e identificar su posición personal.

- Soluciones Operacionales: incluyen acciones realizadas dentro de un alcance organizacional limitado. Incluyen:

  - Entrevista Correctiva: La gerencia se entrevista con la persona involucrada, explica el impacto de su comportamiento. El propósito de la entrevista es generar conciencia y alcanzar un acuerdo de como cambiar el comportamiento.

- Relocación: Recomendar a la persona la búsqueda de trabajo en consultoras de RRHH, de forma de ayudarlo a salir elegantemente de situación dificultosa en la organización

- Soluciones estratégicas: estas acciones son de largo plazo y tienen un alcance organizacional más amplio tales como:

  - Grupo de apoyo: Grupo de referencia, a los cuales la gerencia puede solicitar apoyo para cada una de las personas que presenta el comportamiento no deseado, o grupos de trabajo en donde todas las personas puedan intercambiar los mismos comportamientos y personalidades. El trabajo en este tipo de grupos puede generar la autoconciencia y la mejora personal.
  - Departamento vacío: Los gerentes que presentan estos comportamientos, puede ser realojados en departamentos en donde ellos son los únicos empleados. Estos gerentes usualmente captan el mensaje y tienden a buscar trabajo y continuar sus carreras en otras organizaciones.
  - Reducción forzosa: En algunos casos no existe otro recurso que prescindir de estas personas en el equipo de trabajo u organización.

2.6.4 Irrational Management

Nombre: Irrational Management.
Escala mas frecuente: Empresa.
Nombre de la solución refactorizada: Rational Decisión Making.
Tipo de la solución refactorizada: Rol y Proceso.
Causa Raíz: Responsabilidad.
Fuerzas desbalanceadas: Gestión de recursos.

Síntomas y consecuencias:
El gerente de proyectos carece de la habilidad de tomar decisiones en situaciones críticas para el proyecto y para dirigir al equipo de trabajo, no reconociendo las capacidades del equipo, sus fuerzas y debilidades. No provee de objetivos claros a su equipo y no realiza una comunicación efectiva con ellos. Dentro de los primeros síntomas que encontramos en este antipatrón es el debate continuo sobre un determinado tema, que ante la falta de decisión, no permite al equipo avanzar. Esto genera varias consecuencias, entre ellas:

- Incremento en la frustración del equipo.
- Demora incremental en la entrega del producto.
- La explotación de la situación por los Corncobs.

Causas típicas:
El gerente carece de la habilidad para gestionar:

- Desarrollo de los miembros del equipo.
- A otros gerentes.
- Al proceso de desarrollo.

El gerente no tiene una visión y estrategia clara, lo que deriva en que:

- No puede tomar decisiones.
- Tiene temor al éxito.
- Desconoce el real estado de las actividades del proyecto y de sus entregables.

Solución refactorizada:
Las soluciones para este antipatrón se basan en los siguientes puntos:

- Admitir que se tiene un problema y obtener ayuda: Cuando un gerente sufre de algunas de estas causas, debe primero reconocer que tiene un problema. Asumir lo anterior es el primer paso para identificar los indicadores claves. El gerente irracional se enfrenta al problema una vez que ha alcanzado la etapa de crisis y en estos puntos nadie está deseoso de tomar un problema de estas características y discutirlo. El gerente debe lograr rodearse de un equipo talentoso y acostumbrado a gestionar dentro de un entorno de complejidad.

- Comprender y conocer al equipo de desarrollo: El gerente debe conocer las habilidades técnicas y no técnicas de su equipo. Conocer las primeras le permitirá delegar correctamente el trabajo desde el punto de vista técnico, o formar equipos multidisciplinarios de acorde a las necesidades del proyecto. Conocer las segundas, le ayudará a generar y asignar actividades con componentes de integración interpersonal a los integrantes del equipo.

Otros de los puntos en donde el gerente de proyecto debe de trabajar en función de refactorizar la solución son:

- Dar objetivos claros, y de corto alcance.
- Compartir el foco y la visión del proyecto con todos los integrantes del equipo.
- Buscar y fomentar un proceso de mejora para el proyecto y el equipo de trabajo.
- Facilitar la comunicación dentro del equipo de trabajo y entre sus pares.
- Gestionar los mecanismos de comunicación dentro del proyecto. Los sistemas de e-mail son mecanismos de comunicación muy útiles, pero si son mal utilizados pueden degenerar en una mala comunicación, en insatisfacción del cliente en términos de respuesta, en debates técnicos sin fin. El gerente de proyecto debe realizar el seguimiento de los e-mails dentro del proyecto e identificar este tipo de comunicaciones de forma de poder hablar directamente con los involucrados y restablecer la buena comunicación, proponiendo reuniones en donde llegar a soluciones en forma más rápida y efectiva.
- Gestionar por excepción.
- Aplicar técnicas de toma de decisiones efectivas.

2.6.5 Project Mismanagement

Nombre: Project Mismanagement.
Escala mas frecuente: Empresa.
Nombre de la solución refactorizada: Risk Management.
Tipo de la solución refactorizada: Proceso y Rol.
Causa Raíz: Responsabilidad.
Fuerzas desbalanceadas: Gestión del riesgo.

Síntomas y consecuencias:

Los síntomas y consecuencias de este antipatrón son identificados por los siguientes puntos:

- El diseño es complejo de implementar debido a una carencia en la estrategia de la arquitectura.
- Las revisiones de código e inspecciones suceden en forma no frecuente y agregan un valor mínimo al desarrollo.
- El diseño del test requiere de un esfuerzo extra debido a que no se cuentan con los artefactos adecuados para guíar la actividad de test.
- En las fases de integración y de test existe mucho retrabajo debido al gran número de defectos que deberían haber sido eliminados en las etapas tempranas tales como arquitectura, diseño, inspección, y test unitario.
- El reporte de defectos escala hacia el fin del desarrollo y para las etapas de aceptación y entrega.

Causas típicas:

Están enumeradas como:

- Una arquitectura inadecuada no define los criterios técnicos para las actividades de test, integración, e interoperabilidad.
- La revisión e inspecciones de código, no evalúan los defectos de diseño, las cuales son dirigidas a fases de mayor costo como las etapas de testing.
- Los casos de test tienen en cuenta las necesidades básicas de integración pero no guían en forma completa las necesidades de interoperabilidad de toda la solución.
- Gestión del riesgo no efectiva.

Solución refactorizada:

Realizar una gestión de riesgo apropiada es una manera efectiva de resolver en forma predictiva los síntomas y consecuencias del antipatrón.

Los riesgos son categorizados en diferentes formas útiles: de gestión, puntos de falla comunes en proyectos,y de calidad [MOY89].

Es necesario entender estas categorías de forma de poder calificar los riesgos de mejor forma:

- De gestión: Estos riesgos son principalmente causados y resueltos por el gerenciamiento a nivel corporativo:
    - Procesos: una definición del modelo del ciclo de vida del producto o del servicio a desarrollar.
    - Roles: Responsabilidades específicas para la implementación de los procesos.
- Puntos de falla común en los proyectos: Los riesgos clave del proyecto tienen su base en un conjunto de factores que manejan el riesgo:
    - Sobre presupuesto.
    - Finalización prematura del proyecto.
    - Desarrollo del producto equivocado, que no satisface las expectativas del cliente.
    - Fallas técnicas.
- Calidad:
    - Gestión del proyecto y programa. La efectividad en la planificación y en el control del proceso.

- Identificación de producto: la precisión en la definición del problema.
- Definición de la arquitectura: la especificación de la estrategia de diseño y su codificación.
- Diseño de la solución: la habilidad para entregar la especificación del código en forma optimizada y consistente que soporte la solución y permita su evolución y mantenibilidad.
- Implementación de la solución: la habilidad para producir una implementación de código para el diseño definido.
- Verificación de la solución: las pruebas de que la solución implementada cumple con la especificación del problema
- Validación de la solución: las pruebas de que la solución implementada cumple con lo solicitado por el cliente.
- Soporte del producto: la habilidad para mantener y evolucionar el producto liberado.

Visión común: su ausencia resulta de una vista inconsistente, que deriva en el riesgo de que la solución no satisfaga el alcance de los requerimientos del problema.

Todos los integrantes del proyecto deben compartir y comprender los requerimientos del problema y la solución a la que se desea arribar. La falta de comprensión lleva a una o más fallas:

- Funcionalidades perdidas.
- Funcionalidades incorrectas.
- Sobre funcionalidad.
- Fallas de integración e interoperabilidad.

2.6.6 Micro Management

Nombre: Micro Management.
Escala mas frecuente: Aplicación.
Nombre de la solución refactorizada: Macro Management.
Tipo de la solución refactorizada: Rol.
Causa Raíz: Orgullo e Ignorancia.
Fuerzas desbalanceadas: Gestión de recursos.

Síntomas y consecuencias:
Un gerente de proyectos debe tener las habilidades necesarias para trabajar con las personas, la tecnología y los procesos, así como también la habilidad para lograr el equilibrio necesario entre estos. A menudo, los gerentes de proyectos tienden a sobre gestionar estos aspectos, ya sea por las propias carencias que tienen sobre estos puntos o por el simple hecho de creer que si se hace mayor foco en ellos se mitigarán los riesgos asociados.

El primer síntoma de este antipatrón, es el efecto causado por la falta de foco y control balanceado que tiene el gerente de proyecto en las personas, la tecnología y los procesos, lo cual usualmente desemboca en una carencia de calidad y demora en la entrega del producto. El subsíntoma siguiente es el micromanagement como respuesta a los problemas exhibidos. Los problemas originales usualmente se ven incrementados y la consecuencia final puede ser el desgaste del equipo de trabajo y los efectos laterales de

fallas en el proyecto. Los síntomas primarios con sus consecuencias relacionadas son los definidos por [MOY89] y resumidas en la siguiente tabla:

Síntoma Principal	Consecuencia Principal	Síntoma Secundario	Consecuencia Final
Falla en gestión de personas	Desarrollo del producto equivocado Sobrecosto	Micromanagement	Desgaste del equipo Finalización prematura del proyecto
Falla en gestión de tecnología	Falla tecnológica Sobrecosto		
Falla en gestión de procesos	Desarrollo del producto equivocado Sobrecosto		

Tabla 6- Grado de Impacto

Causas típicas:
La causa de este antipatrón es la visión que tiene el gerente de proyecto, que ve el micromanagement como la solución a los problemas generados con las personas, la tecnología y los procesos. El micromanagement tiene varias formas y puede variar significativamente desde una situación crítica en donde se da ocasionalmente el micromanagement, hasta la gestión diaria de las actividades a realizar por los integrantes del equipo. Un gerente de proyectos usualmente realiza micromanagement debido a su carencia de experiencia y habilidades de gestión, y en un intento por cubrirlas, su respuesta es la de gestionar ajustadamente y controlar todos los detalles que usualmente se delegan a los líderes de equipo o desarrolladores seniors. El micromanagement usualmente tiene un foco limitado en aspectos relacionados a las personas, la tecnología y los procesos, perdiendo de vista el resto de las áreas, lo que causa un incremento del micromanagement en aquellas que no están bajo su control, aun cuando el tipo de control no es el correcto. Cuanto más grande y extenso en el tiempo sea el micromanagement, mayor es el impacto negativo en el desarrollo del software.

Solución refactorizada:
La solución está descrita en base a las siguientes actividades:

- Gestión de las personas:
  El éxito del desarrollo de software depende de que todas las personas trabajen como un equipo y compartiendo los mismos objetivos.

  - Formar y desarrollar a los equipos de trabajo. En lugar de esperar a que el propio equipo se forme solo, es imprescindible que el gerente de proyecto ayude a su formación.

  - Involucrar a todo el equipo en la actividad de planificación. La generación y actualización del plan de trabajo debe involucrar a todos los integrantes del equipo. Esto ayuda a resolver los siguientes problemas:

    - Pobre motivación.
    - Expectativas no realistas.
    - Constante presión con las entregas.
    - Agregar miembros al equipo para aumentar la velocidad de la entrega.

- Gestión de la tecnología:

- Realizar Evaluaciones de tecnología en forma temprana. Esto permitirá resolver los siguientes problemas:
  - Elección de tecnología no familiar.
  - Confiar en productos de software no estables.

- Adoptar herramientas de desarrollo amigables.

- Gestión de Procesos
  Realizar la gestión y el control del proyecto en forma pragmática. El plan de trabajo tiene que ser realizado en forma práctica, estableciendo los hitos principales para cada una de las etapas. Al menos una vez por semana se debe realizar la actualización de dicho plan. Utilizar las luces de un semáforo para clasificar el estado del desarrollo de los componentes del producto y disparar las acciones de mitigación para aquellas que están en amarillo. Para escenarios que requieran un mayor control del riesgo, realizar reuniones de equipo en forma diaria, para actualizar el plan y encontrar los inconvenientes que impidan avanzar al equipo de trabajo y disparar los mecanismos necesarios en forma temprana. Estas acciones permiten realizar un mejor seguimiento al tener la información actualizada y realizar una mitigación de los riesgos en forma proactiva. Además de resolver los siguientes problemas:

  - Falta de un ciclo de planificación.
  - Ciclo de vida inapropiado.
  - Estimación optimista.
  - Calendarios ajustados.
  - Carencia de entregas regulares e incrementales.
  - Parálisis en alguna de las etapas.
  - Carencia de control de calidad.
  - Carencia de control del alcance.

2.6.7 Conclusiones

La lista presentada en esta sección no es completa, y puede ser profundizada en [APAT], [BMMM98], [BMMM00], [KHA05], [LAP06], [AEDW06]. Se presentan aquellos que a mi criterio e interés encuentro elementales para la generación de un lenguaje de patrones y antipatrones que esté orientado a la gestión de proyectos, y que están relacionados directamente al rol de gerente de proyecto.

3 Caso de Estudio

"Lo que tenemos que aprender lo aprendemos haciéndolo."
Aristóteles
(384 a. C. – 322 a. C.)

En este capítulo se presenta el caso de estudio seleccionado con el objetivo de identificar alguno de los patrones y antipatrones de gestión enumerados en el capítulo anterior. Se presenta la propuesta, la descripción del problema basado en datos de proyectos reales y el método de solución para la identificación de patrones. Se describe el prototipo y la arquitectura propuesta y finalmente se presentan los resultados obtenidos para la identificación de los patrones y antipatrones seleccionados.

3.1 Introducción

Las actividades relacionadas con la comunicación y la coordinación son de las más importantes en la gestión de un proyecto. Debido a los modos naturales de comunicación directos e informales existentes entre los integrantes de un equipo, en ciertas ocasiones estas actividades resultan difíciles de observar y también de estudiar en los diferentes tipos de proyectos. La complejidad aumenta cuando lo que se desea observar es la comunicación y coordinación existente en un portafolio de proyectos donde el número de personas y equipos que interactúan es mucho mayor, y en donde las personas pueden participar en más de un proyecto a la vez. Estas actividades se ven afectadas por el ambiente y la cultura organizacional sobre la cual están basadas.

La jerarquía en una empresa de IT tradicional la podemos visualizar como una estructura en donde el CIO se encuentra en el tope de la organización y por debajo se encuentran los integrantes del directorio y luego en otro nivel, se ubican los gerentes. Dependiendo del tipo de organización, a partir del nivel anterior encontramos diferentes formas de estructuras, por ejemplo formas matriciales, funcionales o por proyectos en donde cada una de las personas reporta a su nivel superior.

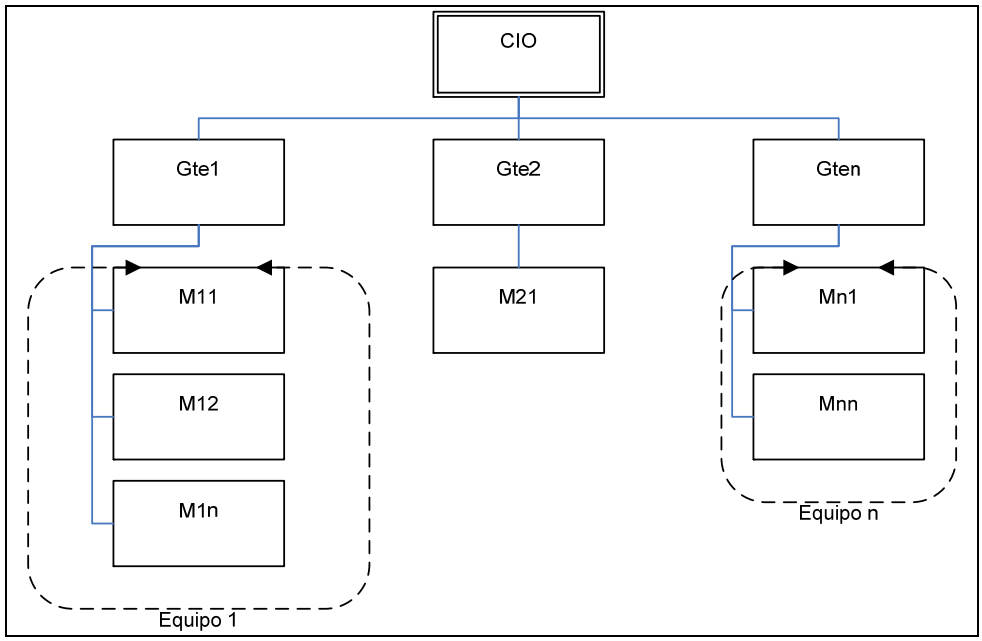


Figura 4 - Jerarquía en organizaciones de IT

Cuanto más grandes son los equipos de trabajo de un proyecto, más aumentan las relaciones interpersonales que se dan dentro de su contexto, y por lo tanto las actividades relacionadas con la comunicación y coordinación crecen de la misma forma. Por ejemplo, si consideramos a *n* como el número de personas que trabajan en un equipo de proyecto, el número de interrelaciones personales crecerá exponencialmente en función de *n*(n-1)/2*.

Si ahora consideramos un equipo de proyecto formado por 3 integrantes, un gerente de proyecto, un analista y 1 programador, el número posible de interrelaciones personales que se da dentro de un proyecto es de 3*(3-1)/2 = 3. Si posteriormente agregamos 1 programador al equipo y luego otro, este número crece primero a 6 y luego a 10.

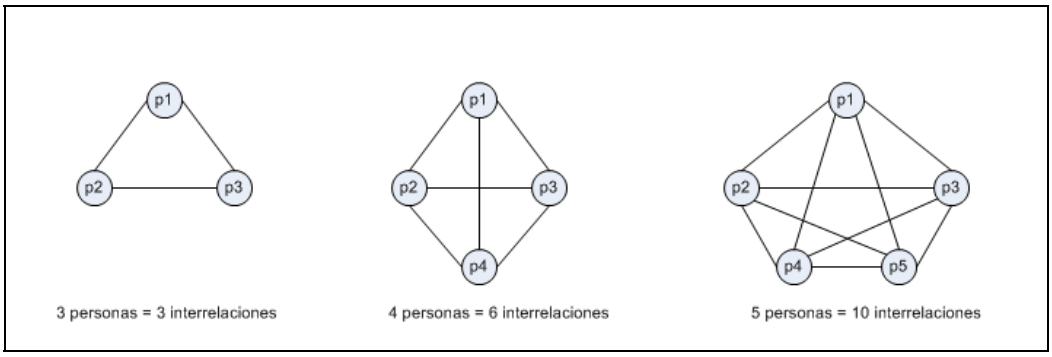


Figura 5 - Interrelaciones según n*(n-1)/2 para equipos de trabajo de 3,4 y 5 personas.

Actualmente la forma de comunicación que se da dentro de las organizaciones ha cambiado. Las organizaciones siguen manteniendo una estructura jerárquica como representación de estructuras de autoridad y de distribución de niveles de responsabilidad, pero el nivel de reporte, comunicación y coordinación ha evolucionado a estructuras de redes que responden a entornos dinámicos que se dan dentro de estas.
En estos entornos, las comunicaciones que suceden dentro de las organizaciones y de los proyectos inmersos en su contexto, han evolucionado por encima de las estructuras jerarquías como forma de responder de manera más flexible a los objetivos, a cambios organizacionales frecuentes y a los cambios de los propios proyectos, pero sin que lo anterior signifique una sustitución de las jerarquías.
En la figura 6 se observa un ejemplo de las interrelaciones de comunicación que se pueden dar a nivel de trabajo entre los diferentes integrantes de una organización en base a una posible estructura jerárquica tal como es representada en la figura 4. En ella se puede apreciar la existencia de interrelaciones que son necesarias para cumplir con los objetivos a nivel de la organización y de sus proyectos asociados pero que no siguen la estructura jerárquica.

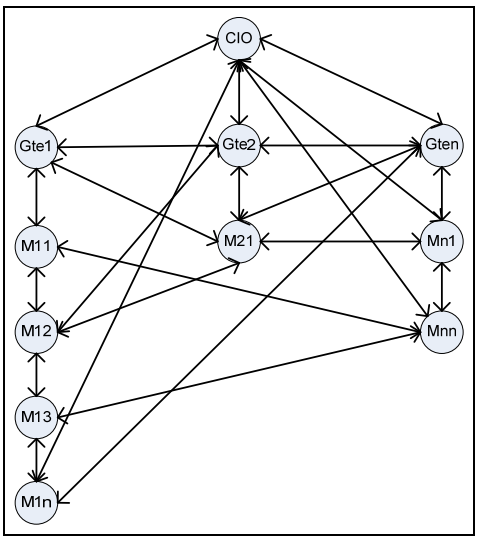


Figura 6 - Ej. De interrelaciones de comunicación para la estructura jerárquica de la figura 4.

Al nivel de un proyecto de software, las comunicaciones se realizan a través de diferentes herramientas de mensajería, en donde el e-mail es el mecanismo que predomina en la comunicación formal entre los equipos de trabajo y el cliente. Por este motivo los archivos de e-mail son una fuente valiosa y útil para realizar el seguimiento de las actividades de comunicación y de coordinación entre los participantes de un equipo de trabajo de un proyecto. Dicho volumen de información se presta a ser estudiado y analizado con el objetivo de encontrar e identificar los comportamientos no deseados en las comunicaciones, y que sirvan como factores de decisión para los gerentes de proyectos. Estudios de este tipo pueden ser realizados utilizando técnicas de data mining, para el descubrimiento de patrones y pueden servir para construir una red social de trabajo que responda a diversas preguntas descubriendo patrones y antipatrones en la comunicación realizada durante el ciclo de vida de un proyecto.

Existen varias razones positivas para que una organización realice una gestión de los e-mails como parte de sus procedimientos de gestión de activos. El contenido capturado en los e-mail corporativos puede ser utilizado como parte de lo que algunas compañías llaman ECM (Enterprise content management) [GRA06], por ejemplo para acelerar la toma de decisiones de negocios y para fortalecer sus procesos de negocios. Gran parte de la información que es de gran utilidad para la inteligencia de negocios está oculta en los sistemas corporativos de e-mail y en las relaciones de comunicación que se dan entre los integrantes de la organización. Esta información puede ser parte de sistemas de gestión de las organizaciones, como por ejemplos los sistemas de CRM (Customer Relationship Management) y ERP (Enterprise resource planning).

La mayoría de las organizaciones [GRA06] manejan sus repositorios de e-mails y de archivos asociados como repositorios para referencias personales. Debido a que la mayoría de las empresas no tienen herramientas que permitan el acceso de un usuario a los e-mails y archivos de otro usuario, las organizaciones resuelven el problema creando diversas cuentas de distribución, en donde cualquier comunicación de carácter no personal y que esté relacionada a la comunicacion de objetivos organizacionales o de proyectos debe estar asociada a esta cuenta. De esta forma, las comunicaciones quedan categorizadas según la cuenta de distribución asociada y bajo la responsabilidad de los usuarios de asociar los mensajes a las cuentas correspondientes. Tanto los participantes del proyecto como los diferentes interesados en el proyecto pueden establecer sus comunicaciones y realizar la coordinación de las actividades involucradas a través de estas cuentas de e-mail, generando la traza de la comunicación disponible para el posterior análisis de información que sea requerido por la organización.

Los gerentes de proyectos pasan gran parte de su horario de trabajo navegando a través de un conjunto de e-mails que pertenecen al proyecto o a los proyectos del portafolio en los cuales participa. El seguimiento de la comunicación y la coordinación de un proyecto a través de e-mails son tareas que requieren diferentes niveles de atención, y que dependen del tipo de proyecto y el estado en el cual este se encuentre. El gerente debe ocuparse de cuidar la comunicación interna y externa, pues el nivel de la comunicación junto con la coordinación del proyecto que es mantenido con el cliente durante la ejecución es uno de los factores de éxito del proyecto.

Es de gran importancia contar con herramientas que permitan tomar decisiones en base al grado de comunicación y de la coordinación que se da internamente con el equipo y su comunicación con el cliente. Los gerentes no deberían basar sus decisiones considerando

únicamente la información relacionada con el costo o al estado de un plan. El nivel de comunicación es un factor importante que en determinadas situaciones sirve para explicar los anteriores y apoyar las decisiones que deban ser tomadas. Una de las soluciones refactorizadas que presenta el antipatrón Irrational Management es la de gestionar los mecanismos de comunicación dentro del proyecto, y contar con herramientas de este tipo sirve a tales efectos. También sirve para detectar las causas que derivan en el antipatrón MicroManagement, analizando las comunicaciones entre el gerente de proyecto y los integrantes del equipo, y descubriendo sus interrelaciones.

3.2 Propuesta

El objetivo de la investigación para el caso de estudio es detectar patrones y antipatrones, especialmente estudiando las relaciones existentes en las comunicaciones de un proyecto. El análisis es realizado a partir de los e-mails pertenecientes a proyectos, apoyando a las soluciones refactorizadas de los antipatrones Irrational Management y MicroManagement vistos en el capítulo 2.

Analizamos el nivel de actividad de los integrantes del equipo de proyecto y sus interacciones, en donde particularmente estamos interesados en responder las siguientes preguntas:

- ¿Qué nivel de MicroManagement existe en el equipo de proyecto?
- ¿En qué etapa del proyecto se realiza el mayor nivel de comunicación entre el equipo y el cliente?
- ¿Qué cantidad de mensajes son enviados y recibidos por cada uno de los integrantes de un proyecto durante el ciclo de vida del proyecto? ¿Cuál es la participación de los diferentes roles?
- ¿Los miembros del equipo que envían mayor número de mensajes dentro del equipo, son lo que tienen mayor comunicación con el equipo del cliente?
- ¿Los miembros del equipo de proyecto que reciben mayor número de mensajes del equipo, son los que tienen mayor comunicación con el equipo del cliente?

Estos tipos de preguntas pueden ser respondidas por sistemas de base de datos actuales utilizando consultas SQL, y dando respuestas concretas. Usando técnicas de data mining, se puede descubrir cuales son los factores que afectan los volúmenes de mensajes intercambiados, descubriendo relaciones y patrones ocultos que pueden no ser obvios.

Para dar respuestas a estas preguntas se requiere ejecutar un proceso de extracción de información desde los repositorios de e-mails correspondientes al proyecto, realizar las transformaciones que sean necesarias y la carga de los datos en un sistema sobre el cual se pueda analizar la información obtenida.

3.3 Descripción del problema

El caso de estudio planteado y desarrollado en el informe es realizado en base a la información real obtenida de proyectos de software y de mantenimiento realizados durante los 6 primeros meses del año 2008. Los proyectos son ejecutados en una fábrica de software local dedicada al desarrollo e implementación de productos y soluciones de software a medida.

Todos los proyectos son realizados para diferentes clientes, los cuales están distribuidos geográficamente dentro y fuera del Uruguay, utilizando los mismos procesos organizacionales. De esta forma los proyectos son ejecutados y tratados en forma uniforme en el contexto de un mismo ambiente cultural y organizacional. Sobre esta base, el análisis de estos dos factores, el cual pudiera derivar en el estudio de los antipatrones asociados, no forma parte del problema a resolver.

Los proyectos utilizan diversas tecnologías de comunicación y colaboración, siendo el medio de comunicación mayormente utilizado durante la ejecución del proyecto el e-mail. En forma periódica se utiliza la videoconferencia como mecanismo de comunicación, lo que posteriormente deriva en la elaboración de un e-mail a todos los participantes resumiendo brevemente los puntos discutidos y las acciones a seguir por los equipos participantes.

Durante el periodo analizado se da la particularidad de que ninguno de los participantes de los equipos de proyecto cambia de rol. Por ejemplo los programadores asignados al proyecto siguen ocupando los mismos roles y no pasan a ser analistas o especialistas técnicos. En el caso de los clientes se produce la misma situación, durante el periodo analizado los clientes continúan cumpliendo los mismos roles que al inicio del periodo de estudio. En determinados proyectos, algunas personas son reasignadas, pasando de un proyecto a otro proyecto, pero manteniendo el mismo rol.

Por definición, todas las comunicaciones entre los participantes de los diferentes proyectos son copiadas a casillas de distribución creadas específicamente a los efectos de asociarles todos los mensajes correspondientes a los proyectos. Las comunicaciones internas de los equipos del proyecto se copian a estas casillas, y las comunicaciones desde y hacia el cliente también siguen esta regla. Esto permite a todos los miembros del equipo tener acceso a la información generada durante el periodo analizado del proyecto.

Las únicas personas que no tienen acceso a esta información son las pertenecientes al equipo del cliente.

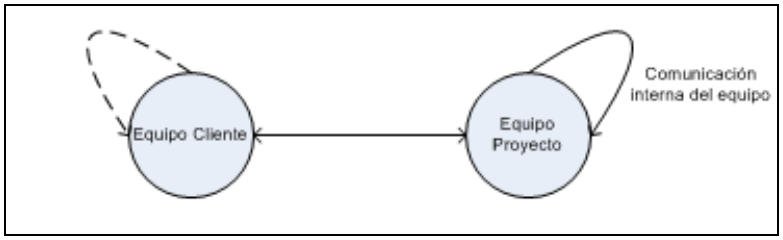


Figura 7 - Canales de comunicación entre equipos del cliente y equipo del proyecto

La comunicación interna del cliente no está incluida dentro de la definición anterior, en algunos casos el propio mensaje incluye información de la traza de la comunicación correspondiente al lado del cliente hasta que el mensaje llega a la lista de distribución, pero esta no es tenida en cuenta en el estudio.

Estos flujos de comunicación son representativos y relevantes en la información asociada a los proyectos estudiados. Forman parte de la base de colaboración de la empresa y permite a todos los integrantes de un proyecto, independientemente de la fase en la cual ingresan al proyecto, tener acceso en forma sencilla a los temas que fueron tratados durante el ciclo de vida, y de forma sencilla tener referencias a respuestas, acuerdos, definiciones, y seguimiento de problemas que se dieron durante el mismo.

3.4 Reglas

El nivel de actividad de los participantes de un proyecto y sus relaciones puede ser medido. Para satisfacer este punto se definen un conjunto de reglas, las que son desconocidas por los integrantes del equipo de proyecto y que sirven para realizar el análisis.

Reglas:

- Todo cliente *Ci* que inicie una conversación y cuyo destino sea alguna persona del equipo de proyecto (cuentas de e-mail, o la propia lista de distribución) debe tener una respuesta de *Mi*, siendo *Mi* integrante del equipo de proyecto.

- Toda respuesta a una conversación inicial de *Ci*, debe tener una respuesta de *Mi* en un tiempo no mayor a *Ti*.

- Toda respuesta de *Ci* a una conversación inicial de *Mi*, no debe ser mayor a *Tii*.

- Toda conversación iniciada por *Mi*, debiera estar finalizada con un mensaje de *Mi*.

- El numero de mensajes de Mi, en una conversación iniciada por algún *Mi*, debe ser mayor al número de mensajes de *Ci*.

- El número de mensajes en una conversación no debe ser mayor a *Nm*.

- El número de mensajes de *Mi* = *GP* (Gerente de Proyecto) enviados a los miembros del equipo debe ser mayor para los roles de mayor jerarquía, Analistas, Arquitectos, Diseñadores, Programadores Senior, Programadores. De darse el caso de que la cantidad de mensaje que envía un GP a un Programador es mayor a las que envía a un Analista, se puede estar en presencia del patrón de MicroManagement.

- El número de mensajes enviados por *Mi* = *GP* a un *Ci* dentro de un periodo de tiempo *Pi*, no debe ser menor a *Nmpi* (numero de mensajes por periodo)

Los casos de aquellos mensajes que son enviados únicamente a la lista de distribución, tienen los siguientes significados:

- Simple olvido de la persona.

- Enviar el resultado de una comunicación directa con el cliente, que no es enviada en su oportunidad con copia a la lista. Las comunicaciones directas con el cliente para tratar algún tema o recordar la atención a temas o acciones acordadas, en algunas ocasiones es realizada sin copia a la casilla para dar la sensación de una conversación individual en la cual participan dos personas o un grupo reducido y no una reunión grupal. La regla indica que esta comunicación debe ser copiada de igual forma en la lista de distribución.

3.5 Método de solución

Para realizar el análisis del problema descrito en la sección anterior y con el objetivo de responder las preguntas que conduzcan a determinar la existencia de los patrones y antipatrones que estamos estudiando, se decide realizar un estudio con apoyo de técnicas de data mining. Por este motivo se siguen las técnicas definidas de acuerdo al estándar CRISP-DM [CRISP01] [CRISP02].

3.5.1 CRISP-DM

El estándar define un conjunto de 6 fases involucradas en el proceso de desarrollo. Fases que van desde la compresión del negocio hasta la última de las fases, el deployment de la solución construida. Estas fases pueden ser que no sean finalizadas en un orden estricto, ya que el proceso normalmente puede involucrar referenciar a una fase anterior y la ejecución repetida de alguna de estas. El proceso debe verse como un proceso de desarrollo iterativo e incremental.

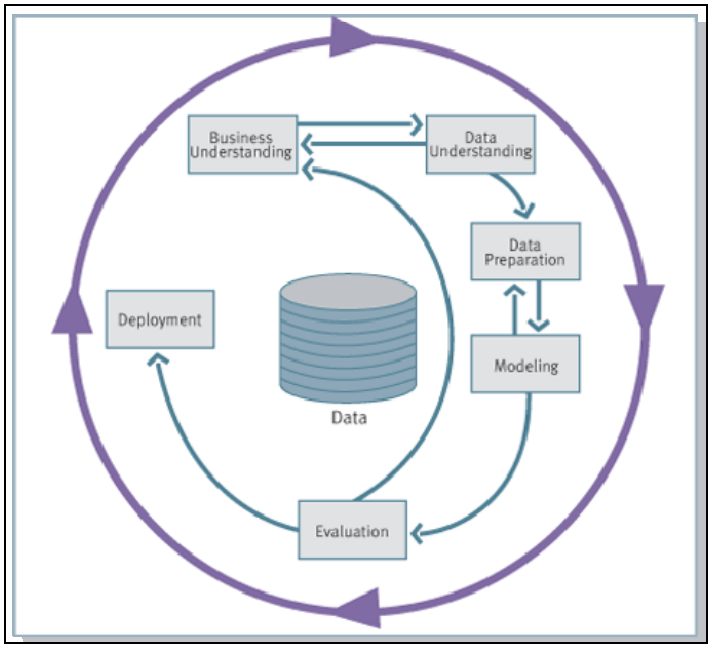


Figura 8 - Proceso CRISP-DM

3.5.2 Business understanding

Esta fase involucra la compresión de los requerimientos de negocio necesarios para la utilización de data mining. Esto permite generar el conocimiento necesario para definir el problema de data mining y determinar un plan preliminar diseñado para satisfacer los objetivos planteados.

En este punto del desarrollo es importante poder responder preguntas del estilo:

a) ¿Qué información se desea encontrar?
b) ¿Buscamos predicciones o solo patrones?
c) ¿Qué resultados queremos producir?
d) ¿Qué relaciones tratamos de encontrar?

En el caso de estudio particular las respuestas que se dan a estas preguntas son:

a) La información que se desea encontrar es la que están relacionadas a las propuestas mencionadas en el punto 3.2.

- ¿Qué nivel de MicroManagement existe en el equipo de proyecto?

  Como se analizó en el antipatrón de Micromanagement, este tiene varias formas de presentarse y algunas de ellas están asociadas a la gestión diaria de cada una de las actividades de los integrantes del proyecto y por parte del gerente de proyecto. Esta información puede ser descubierta al nivel de las comunicaciones que se dan dentro de un proyecto, observando la frecuencia de mensajes desde el gerente de proyecto hacia los integrantes del proyecto y viceversa. Cuando esta comunicación, que por ejemplo se puede dar entre en un gerente de proyecto y un desarrollador, es aislada y frecuente, puede ser un indicador de la existencia de micromanagement. El nivel asociado al micromanagement puede ser determinado en base a la frecuencia que se da en dicha comunicación.

- ¿En qué etapa del proyecto se realiza el mayor nivel de comunicación entre el equipo y el cliente?

  Analizando el flujo de los mensajes enviados y recibidos entre los integrantes del proyecto, y realizando un mapeo con las etapas del proyecto, se puede determinar cuales son las etapas que tienen un mayor nivel de comunicación. Esto ayuda a determinar patrones de comunicaciones discriminados por etapa y que sirven para descubrir la existencia de antipatrones de MicroManagement y de Irrational Management.

  Por ejemplo, detectando hilos de conversaciones sin fin, que podrían llegar generar insatisfacciones de los clientes en términos de la respuesta que es esperada, lo que ayudaría al gerente a identificar el problema de forma de poder hablar directamente con los involucrados, restableciendo la buena comunicación y proponiendo mecanismos para llegar a soluciones en forma más rápida y efectiva.

  - ¿Qué cantidad de mensajes son enviados y recibidos por cada uno de los integrantes de un proyecto durante el ciclo de vida del proyecto? ¿Cuál es la participación de los diferentes roles?

El análisis de los mensajes ayuda a determinar la participación de los gerentes, analistas y desarrolladores y sus relaciones en las diferentes etapas del ciclo de vida de un proyecto. Así mismo puede ayudar a determinar las etapas en las cuales los interesados en el proyecto participan con mayor frecuencia.

- ¿Los miembros del equipo que envían mayor número de mensajes dentro del equipo, son lo que tienen mayor comunicación con el equipo del cliente?

- ¿Los miembros del equipo de proyecto que reciben mayor número de mensajes del equipo, son los que tienen mayor comunicación con el equipo del cliente?

  Las respuestas a las dos preguntas anteriores, puede ayudar a detectar líderes o referentes técnicos dentro del equipo, que a veces por la propia personalidad introvertida de los integrantes del equipo pasen desapercibidos en otros niveles de comunicaciones. También puede servir para: detectar corncobs que se manejen a través de la comunicación vía e-mail según lo visto en el antipatrón Corncobs, y apoyar la solución refactorizada de este antipatrón.

b) La respuesta a la segunda pregunta y para el caso de estudio, es la búsqueda de patrones y antipatrones de comunicación dentro del alcance de la propuesta. La predicción en base a la información analizada puede ser objeto de trabajos futuros en la misma área, con el objetivo de mejorar la gestión de los proyectos considerando las experiencias necesarias que ayuden a predecir y prevenir comportamientos antes que los mismos se conviertan en antipatrones.

c) Se desea producir información que sea de ayuda para la toma de decisiones en la gestión de los equipos de proyecto. Información que sirva para detectar las causas de antipatrones y que sirvan como apoyo en las soluciones refactorizadas. Información que sirva para el descubrimiento de nuevos patrones y antipatrones.

d) Las relaciones son las indicadas en las respuestas al punto a.

3.5.3 Data Understanding

En esta fase se recolectan los datos de interés, se examinan y buscan patrones y subconjuntos que permitan una familiarización con los datos del problema, identificando problemas de calidad de datos y determinado su validez. Lo que puede incluso llevar a detectar patrones sin el uso de data mining.

Estructura de Mensajes

De forma de analizar los datos a considerar en el caso de estudio, es necesario comprender el formato de las estructuras de los mensajes de e-mails, los cuales están estructurados en dos secciones:

- **Header:** Sección estructurada en campos, cada uno con su propio nombre y valor. Los campos más usuales son:
  - From: la dirección e-mail y opcionalmente el nombre del remitente.
  - To: La o las direcciones de e-mails, opcionalmente el nombre o los nombres de los receptores del mensaje.

- Subject: Breve resumen del contenido del mensaje
- Date: Fecha y hora local del mensaje enviado.

Adicionalmente a estos campos, el header incluye otros según el [RFC5322]:

- Cc: Carbon copy. La o las direcciones de e-mails, opcionalmente el nombre o los nombres de los receptores con copia del mensaje.
- Bcc: Blind Carbon Copy. La o las direcciones de e-mails, opcionalmente el nombre o los nombres de los receptores con copia oculta del mensaje.
- Content-Type: Información relacionada al formato de como un mensaje debe ser desplegado al momento de ser recibido y abierto por el/los receptores. Generalmente está asociado a un tipo MIME.

- **Body:** La que contiene el mensaje en texto desestructurado.

MIME(Multipurpose Internet e-mail Extensions) [MIME01], es un estándar que extiende el formato de e-mail de forma de soportar principalmente:

- Un conjunto de caracteres más amplio que el código ASCII, tanto a nivel del header como del body.
- Adjuntos en formatos diversos

Datos Recolectados

Durante el periodo comprendido entre enero y marzo del 2008 se realiza la recolección de información en base a los headers de los e-mails enviados y recibidos en la lista de distribución de cada uno de los proyectos que se estudian.

- Se estudian un total de 17 proyectos.
- 7789 horas de trabajo son distribuidas entre los proyectos considerando la dimensión presupuestada para cada uno.
- Se contabilizan 6297 mensajes.
- Participan 405 cuentas de e-mail:
  - 62 cuentas de e-mail pertenecientes a la organización de las cuales:
    - 12 pertenecen directamente al equipo de proyecto asignado (generalmente con una estructura de equipo formado por un jefe de proyecto, arquitecto, analista, programador, especialista técnico y tester)
    - 50 son miembros de la organización no directamente vinculados con el proyecto (personas del área comercial, del área administrativa, áreas de servicio)
  - 343 cuentas de e-mail pertenecientes al cliente. Categorizadas en tres niveles: interesados, funcionales y técnicos.

Proyecto	Esfuerzo en horas	Cantidad de Mensajes	# Personas en equipo de proyecto.	#Personas pertenecientes a la organización	#Personas pertenecientes al cliente
Proyecto 1	625	810	5	11	36
Proyecto 2	779	92	4	8	9
Proyecto 3	912	757	4	9	73
Proyecto 4	46	73	3	2	3
Proyecto 5	40	153	5	3	9
Proyecto 6		73	3	8	6
Proyecto 7	859	199	3	5	19
Proyecto 8	838	195	4	10	23

Proyecto 9		107	5	4	7
Proyecto 10	505	324	4	11	5
Proyecto 11	137	116	4	0	3
Proyecto 12	103	177	2	20	7
Proyecto 13	330	122	4	9	5
Proyecto 14	450	590	3	13	8
Proyecto 15	603	275	5	3	10
Proyecto 16	1108	1907	4	8	41
Proyecto 17	454	327	4	3	24

Tabla 7 - Lista de proyectos analizados

3.5.4 Data preparation

En esta fase se selecciona el diseño del origen de datos para el proceso de data mining y se crea el origen. El diseño del origen de datos debe soportar los métodos de data mining a utilizar.

Durante esta fase se extrae, transforma y cargan los datos desde sus orígenes. La preparación de datos es una actividad que se realiza en varias oportunidades y no necesariamente con un orden determinado. El resultado de esta fase es el conjunto de datos final que sirve como entrada al modelo de datos de data mining que se seleccione.

La información asociada a los mensajes relacionada con las cuentas de e-mail correspondientes a las listas de distribución se extrae desde el sistema de e-mail perteneciente a la empresa de software. La información extraída para el análisis se corresponde con los datos contenidos en el header del mensaje, de acuerdo con la estructura explicada en el punto 3.5.3.

Pasos:

1) Durante el proceso de extracción se accede solamente a aquellos tipos de mensajes **Mensaje**, y para cada uno de ellos se obtiene el valor de cada uno de los campos del header. Esta información es cargada en la tabla de la base de datos utilizada a tales efectos. En esta instancia no se realiza ninguna transformación.

Siendo ME la tabla de Mensajes exportados, el algoritmo de extracción se reduce a:

Para cada ld perteneciente al conjunto de listas de distribución
Para cada msg de tipo Mensaje perteneciente a ld
ME.IdMensaje = numerador
ME.Fecha = msg.Fecha
ME.Subject = msg.Subject
ME.SenderE-mailAddress = msg.SenderE-mailAddress
ME.MensajeTo = msg.MensajeTo
….
Fin
Fin

2) El siguiente paso de la extracción es transformar los datos originales en datos que sean manipulables de mejor forma. Este es el caso de:

Origen	Transformación
Fecha (string)	Conversión a datetime.
DistributionList	Es asociada al Identificador de Proyecto.
Tamaño	Conversión a int. Indicando el tamaño del mensaje.
Adjuntos	Conversión a int. Indicando la existencia de archivos adjuntos al mensaje de e-mail

Tabla 8 – Transformación de datos asociados a campos de un mail

3) El siguiente paso es el relacionado con el tratamiento de las direcciones de e-mails asociados:

El campo SenderE-mailAddress contenido en el header de los mensajes sigue el formato LDAP (Protocolo de Acceso Ligero a Directorio) para las cuentas de las personas pertenecientes a la organización.

Origen	Transformación
SenderE-mailAddress Para las cuentas de la organización: /O=*NombreOrganizacion*/OU=*UnidadOrgnizacional*/CN=*Recipients*/CN=*Cuenta*	*cuenta@Nombreorganizacion.com*

Tabla 9 – Transformación de cuentas de email

4) Considerando los valores contenidos en los campos From, To y BCC del header del mensaje, se extrae dicha información para cargar la tabla de personas.

Los campos To y BCC incluyen la lista de cuentas de e-mail de las personas que reciben el mensaje.

Siendo P la tabla de Personas, el algoritmo de extracción se reduce a:

```
USE PrototipoTesis
GO

-- Declaracion de variables que almacenan el resultado del  FETCH.
DECLARE @IdMensaje bigint, @ConversationTopic nchar(500)
DECLARE @SenderName nchar(255), @SenderE-mailAddress nchar(100)
DECLARE @MensajeTo nchar(1000), @MensajeCC nchar(1000)
DECLARE @MiembroMensajeTo nchar(1000), @MiembroMensajeCC nchar(1000)
DECLARE @IdPersonaFrom int, @IdPersona int
DECLARE @Orden int

DECLARE Mensaje_cursor CURSOR FOR
SELECT IdMensaje, ConversationTopic, SenderName, SenderE-mailAddress, MensajeTO,
MensajeCC FROM Mensaje
ORDER BY IDMensaje

OPEN Mensaje_cursor

FETCH NEXT FROM Mensaje_cursor
INTO @IdMensaje, @ConversationTopic, @SenderName, @SenderE-mailAddress, @MensajeTo,
@MensajeCC


-- Check @@FETCH_STATUS to see if there are any more rows to fetch.
WHILE @@FETCH_STATUS = 0
```

```
BEGIN

	Set @SenderE-mailAddress = dbo.TransformarCuentaE-mail(@SenderE-mailAddress)
	Set @IdPersonaFrom = dbo.ObtenerIdPersona(@SenderName,@SenderE-mailAddress)
	Set @Orden = 1
	-- Obtener por separado cada integrante de MensajeTO
	-- Mientras @MensajeTO contenga miembros
	While len(@MensajeTo) > 0
	Begin
		-- @MiembroMensajeTo = el primer miembro (a la izquierda del primer ;)
		-- @MensajeTo = El resto de los miembros (a la derecha del primer ;)
		If PatIndex('%;%',@MensajeTo) <> 0
		Begin
			SET @MiembroMensajeTo = ltrim(rtrim(Left(@MensajeTo,
			PatIndex('%;%',@MensajeTo)-1)))
			SET @MensajeTo = ltrim(rtrim(substring(@MensajeTo,
			PatIndex('%;%',@MensajeTo)+ 1, len(@MensajeTo))))
		End
		Else
		Begin
			SET @MiembroMensajeTo = ltrim(rtrim(@MensajeTo))
			SET @MensajeTo = ''
		End

		--  Actualizar tablas correspondientes segun logica definida
		Execute AgregarPersona @MiembroMensajeTo , @IdPersona OUT
		Execute AgregarRelacionMensaje @IdMensaje, @ConversationTopic,
			 @IdPersonaFrom , 'T', @IdPersona, @Orden
		Set @Orden = @Orden + 1
		-- Fin Actualizacion

	End
	-- Fin

	Set @Orden = 1
	-- Obtener por separado cada integrante de MensajeCC
	-- Mientras @MensajeCC contenga miembros
	While len(@MensajeCC) > 0
	Begin
		-- @MiembroMensajeTo = el primer miembro (a la izquierda del primer ;)
		-- @MensajeTo = El resto de los miembros (a la derecha del primer ;)
		If PatIndex('%;%',@MensajeCC) <> 0
		Begin
			SET @MiembroMensajeCC = ltrim(rtrim(Left(@MensajeCC,
			PatIndex('%;%',@MensajeCC)-1)))
			SET @MensajeCC = ltrim(rtrim(substring(@MensajeCC,
			PatIndex('%;%',@MensajeCC)+ 1, len(@MensajeCC))))
		End
		Else
		Begin
			SET @MiembroMensajeCC = ltrim(rtrim(@MensajeCC))
			SET @MensajeCC = ''
		End

		--  Actualizar tablas correspondientes segun logica definida
		Execute AgregarPersona @MiembroMensajeCC , @IdPersona OUT
		Execute AgregarRelacionMensaje @IdMensaje, @ConversationTopic,
			 @IdPersonaFrom , 'C', @IdPersona, @Orden
		Set @Orden = @Orden + 1

	End
	-- Fin


FETCH NEXT FROM Mensaje_cursor
INTO @IdMensaje, @ConversationTopic, @SenderName, @SenderE-mailAddress,
@MensajeTo,@MensajeCC
END

CLOSE Mensaje_cursor
DEALLOCATE Mensaje_cursor
GO
```

3.5.5 Modelling

Para esta fase se seleccionan y aplican varios métodos de modelado de datos de forma de lograr su optimización. Si se elige un método para el cual no se lo planificó en las fases de Data understanding y Data preparation, es usual que se deba repetir la fase anterior.

3.5.6 Evaluation

En esta fase se evalúan los modelos de data mining para asegurar que satisfacen los problemas de negocio, que los pasos para crear el modelo son correctos y que los resultados son útiles para el estudio bajo análisis.

3.5.7 Deployment

En este paso se documenta y realiza el proceso de deployment.
Después de que han sido creados los modelos, existen un conjunto de tareas adicionales y necesarias para presentar los datos en una forma útil y comprensible.

3.6 Arquitectura del prototipo propuesto

3.6.1 Introducción

El caso de estudio es acompañado por el análisis y diseño de un prototipo, identificado con el nombre SPAGPE (Prototipo de descubrimiento de Patrones&Antipatrones de gestión de proyectos de software basado en e-mails) y que sienta las bases para el desarrollo de trabajos futuros en el área. A continuación se presentan las diferentes secciones que definen la arquitectura inicial del prototipo. Estas secciones tienen como guía el esquema de un documento de representación de arquitectura SAD [SAD].

3.6.2 Propósito

Esta sección del informe tiene como propósito brindar una visión comprensible de la arquitectura general del prototipo SPAGPE, utilizando diferentes vistas de la arquitectura para ilustrar diferentes aspectos del sistema. Captura las decisiones más importantes en lo que respecta a la arquitectura del sistema, obtenida durante la elaboración del prototipo.

Está dirigido a distintos tipos de actores: personas interesadas en conocer la arquitectura del prototipo presentado y sirve como referencia para aquellos interesados en la creación de sistemas relacionados a la temática planteada en este informe.

3.6.3 Alcance

Se profundiza principalmente en las vistas de servicios y vista lógica, en donde se incluyen algunos elementos significativos al resto de las vistas. No se pretende dar una definición completa de la arquitectura sino una introducción que sirva como puntapié inicial a otros trabajos.

3.6.4 Representación de la Arquitectura

El prototipo SPAGPE está concebido como la base para dar respuesta a las interrogantes planteadas en el punto 3.2 del informe y que están relacionados con la detección de patrones y antipatrones de gestión de proyectos.

La arquitectura está representada por diferentes vistas utilizando notación UML [UML03] de forma que se puedan visualizar, entender y razonar sobre los elementos

significativos de la arquitectura e identificar las áreas de riesgo que puedan requerir mayor detalle de elaboración en propuestas futuras. Este documento es una forma de comunicar el modelo del prototipo, presentando la información y discusiones en forma estructurada.

La siguiente sección detalla las vistas de la arquitectura que serán utilizadas para cubrir las dimensiones mencionadas, presentando a continuación el framework arquitectónico utilizado.

Representación

- Vista de Casos de Uso: Describe los procesos de negocio más significativos y el modelo del dominio. Presenta los actores y los casos de uso para el prototipo.

- Vista Lógica: Describe la arquitectura del prototipo presentando varios niveles de refinamiento. Indica los módulos lógicos principales, sus responsabilidades y dependencias.

- Vista de Deployment: Presenta aspectos físicos como topología, infraestructura informática.

Framework Arquitectónico

La arquitectura sigue el framework "4+1" (con variantes) presentado en [Kru95]; este framework define cuatro vistas para la arquitectura (4) en conjunto con los escenarios de uso (1), y es presentado en la siguiente figura:

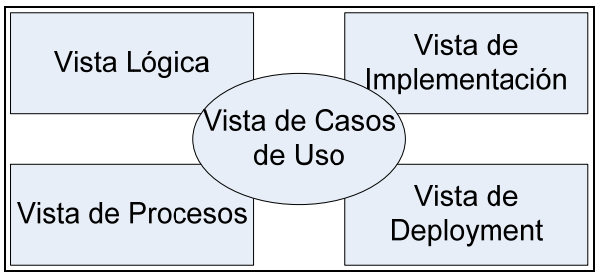


Figura 9 - Modelo 4+1 para representación de arquitecturas de Krutchen

El mapeo de las vistas utilizadas a las propuestas en el framework se presenta en la siguiente tabla:

Framework 4+1	Arquitectura
Use-Case View	Casos de Uso
Logical View	Lógica
Process View	Servicios
Implementation View	Datos
Deployment View	Deployment

Tabla 10 – Mapeo Framework 4+1 con Arquitectura

3.6.5 Vista de Casos de Uso

Esta vista presenta la percepción que tiene el usuario de las funcionalidades del prototipo. Se presentan los procesos de negocio más importantes, los casos de uso críticos que se derivan de éstos. Se muestran sus actores y se detallan los casos de uso significativos que tienen influencia sobre la arquitectura candidata.

Modelo del Dominio

El modelo dominio incluye el vocabulario necesario de comprender desde el punto de vista del problema y de la arquitectura.

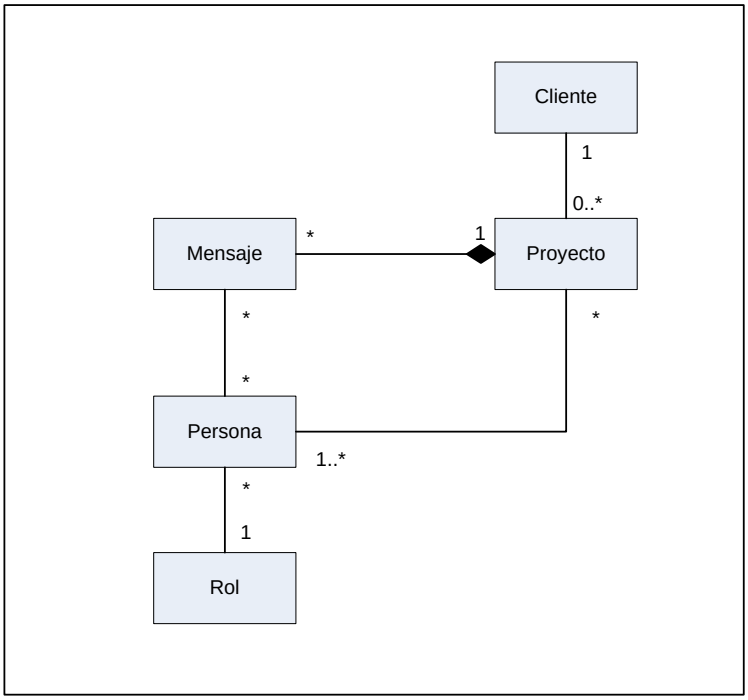


Figura 10 - Modelo de Dominio del prototipo

Actores

Los actores son los que interactúan con el prototipo y son los interesados en la problemática presentada en el informe.

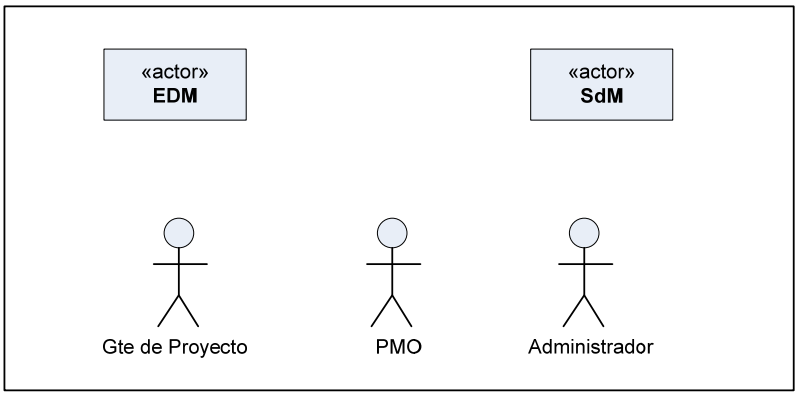


Figura 11 - Actores participantes del prototipo

SdM: Sistema de mensajería, por ejemplo, MS-Exchange server, Lotus Notes entre otros.

EDM: Agente extractor de mensajes desde el **SdM.**

Gte de Proyecto: Gerente de Proyecto.

PMO: Miembro de la oficina de proyectos de la organización.

Administrador: Responsable de la administración de la solución prototipo propuesta.

Los componentes y sistemas que forman parte de la arquitectura del prototipo son expresados en forma canónica.

El Modelo de Casos de Uso del Sistema se organiza en paquetes de casos de usos utilizados en el prototipo SPAGPE.

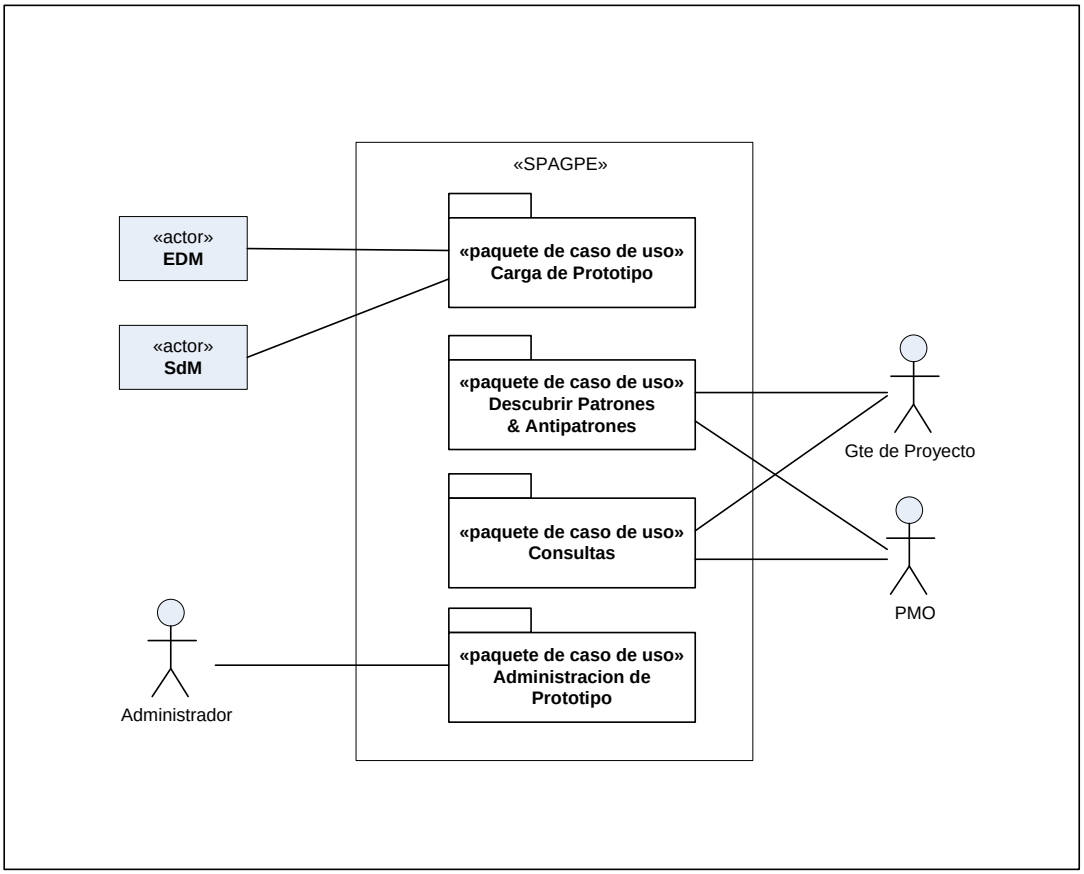


Figura 12 - Modelos de Caso de Usos del prototipo

Se atacan principalmente aquellos casos de uso relevantes para la lógica planteada en el caso de estudio del informe.

Casos de uso

Los casos de uso del sistema son priorizados, se eligen los más importantes, y se deja para trabajos futuros completar el resto de los casos.

CUS – CP.1 ExportarMails

Nombre	CUS – CP.1 ExportarMails
Actores	EDM, Sistema de e-mail, SPAGPE
Sinopsis	Este caso de uso comienza cuando el EDM indica al prototipo SPAGPE que desea realizar la exportación de los e-mails asociados a todas las cuentas de distribución asociados a un proyecto y definidas en el prototipo.
Escenario típico	
	1. EDM se conecta al sistema de e-mail indicando credenciales de usuario. 2. El sistema de e-mail valida las credenciales suministradas. 3. Para cada una de las cuentas de distribución definidas en SPAGPE: a. EDM accede a la cuenta de distribución en el sistema de e-mail. b. Para cada uno de los recursos almacenados en la cuenta de distribución: i. EDM obtienen la información del recurso. ii. Si el tipo de recurso es Mensaje, EDM obtiene las siguientes propiedades del Mensaje: *DistributionList, Fecha, Subject, conversationTopic, SenderName, SenderE-mailAddress, MensajeTo, MensajeCC, Tamaño, Importancia, Adjuntos.* iii. EDM registra esta información en SPAGPE. iv. Sigue en 2.b c. Sigue en 3. 4. Éxito.
Escenarios alternativos	
	2a. La información de credencial no es válida 1. Sistema de e-mail notifica falla de acceso a EDM. SPAGPE deja registro del fallo. 2. Fallo 3a. Las listas de distribución no son válidas en el sistema de e-mail 1. EDM no puede acceder a lista y notifica al SPAGPE. 2. EDM continúa en 3.

CUS – CP.2 ExportarMailsporProyecto

Nombre	CUS – CP.2 ExportarMailsporProyecto
Actores	EDM, Sistema de e-mail, SPAGPE
Sinopsis	Este caso de uso comienza cuando el EDM indica al prototipo SPAGPE que desea realizar la exportación de los e-mails de una cuenta de distribución asociada a un proyecto, y definida en el prototipo.
Escenario típico	
	1. EDM se conecta al sistema de e-mail indicando credenciales de usuario. 2. El sistema de e-mail valida las credenciales suministradas. 3. EDM accede a la cuenta de distribución en el sistema de e-mail. a. Para cada uno de los recursos almacenados en la cuenta de distribución: i. EDM obtienen la información del recurso. ii. Si el tipo de recurso es Mensaje, EDM obtiene las siguientes propiedades del Mensaje: *DistributionList, Fecha, Subject, conversationTopic, SenderName, SenderE-mailAddress, MensajeTo, MensajeCC, Tamaño, Importancia, Adjuntos.* iii. EDM registra esta información en SPAGPE. b. Sigue en 3. 4. Éxito.
Escenarios alternativos	
	2a. La información de credencial no es válida

	3. Sistema de e-mail notifica falla de acceso a EDM. SPAGPE deja registro del fallo. 4. Fallo 3a. Las listas de distribución no es válidas en el sistema de e-mail 3. EDM no puede acceder a lista y notifica al SPAGPE. 4. EDM continúa en 3.

3.6.6 Vista Lógica

En este punto se presenta la vista lógica de la arquitectura a través de un refinamiento partiendo desde un nivel de abstracción mayor a uno menor. Como premisa la definición del prototipo está guiada por el patrón de arquitectura de orientación a servicios. Este patrón propone una definición de servicios independientes que ofrecen un conjunto de funcionalidades que son utilizadas por los diferentes actores de una forma altamente desacoplada, y con interfaces bien definidas.

SPAGPE

En base a lo anterior, la representación de la arquitectura para el prototipo SPAGPE utilizando el patrón de SOA, es el que se muestra en la figura.

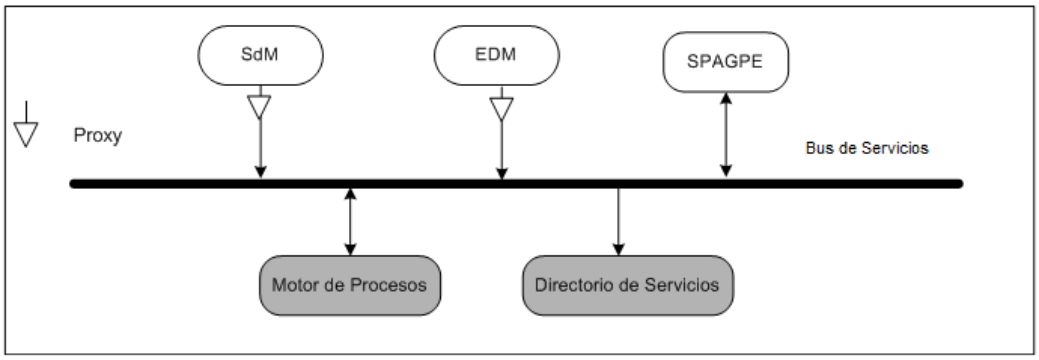


Figura 13 - Patrón de Arquitectura para el prototipo

Todos los actores identificados participan de una arquitectura orientada a servicios, en donde se exponen a través del directorio de servicios cada uno de los servicios provistos.

El directorio de servicios es un proveedor de servicios que está encargado de almacenar los contratos de los servicios publicados por los proveedores para que cada actor los pueda consumir, logrando de esta forma un bajo acoplamiento entre proveedores y consumidores.

El bus de servicios es responsable de la comunicación entre los diferentes participantes del sistema, y de la administración de las interacciones entre los consumidores y proveedores, agregando funcionalidades de mayor nivel como puede ser la seguridad y mecanismo de transaccionalidad.

En la representación de la arquitectura se aprecia la inclusión de lo que se denomina motor de procesos, el cual en un contexto de reglas de negocios complejas para la determinación de patrones y antipatrones, permite definir sus procesos asociados y facilitar la resolución de las reglas.

Refinando el propio prototipo SPAGPE, se plantea de usar el patrón Layers para su representación.

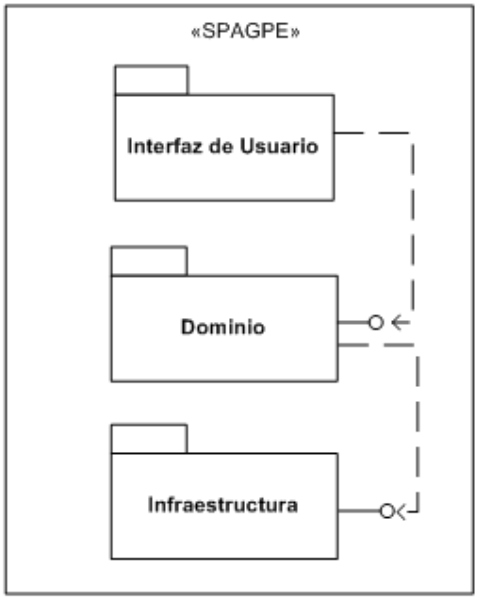


Figura 14 - Patrón de Layers para el prototipo

Cada capa del patrón identifica los diferentes componentes necesarios para la solución al problema planteado.

- **Interfase de usuario**
  i. Organizada mediante el patrón MVC – Model-View-Controller [MVC]

- **Dominio**
  i. Componente Proyecto
  ii. Componente Mensajes
  iii. Componente Patrones & Antipatrones
  iv. Componente Auditoria
  v. Componente Seguridad

- **Infraestructura**
  i. Componentes de acceso a datos

Para el monitoreo de SPAGPE se plantea el uso del patrón cliente – servidor [BUSCH96], en donde un monitor oficia de cliente, débilmente acoplado al SPAGPE, que en este caso tiene el rol de Servidor. El bajo acoplamiento se logra usando un broker [BUSCH96], representando por el directorio de servicios.

Sistema EDM

El sistema EDM, responsable de la extracción de información de e-mail desde el sistema de e-mail de la organización, transformación y carga en el sistema SPAGPE fue plateado con el patrón de arquitectura Pipes & Filters.

EDM debe extraer y procesar los mensajes del sistema de e-mails, teniendo la flexibilidad necesaria para permitir el ordenamiento de los pasos considerados en la transformación de los datos del e-mail.

Este patrón nos permite a futuro realizar mejoras cambiando los pasos de procesamiento o recombinándolos e incluso combinando diferentes orígenes de datos (por ejemplo diferentes sistemas de e-mails).
La solución se basa en dividir las tareas de extracción, transformación y carga en diferentes pasos de procesamiento, conectados por el flujo de datos.

Cada paso es realizado mediante un filter, en donde este consume y produce datos de forma incremental, en lugar de consumir toda su entrada antes de producir la salida.
Los datos que se consumen a partir de una lista de distribución perteneciente a un proyecto, son los que definen un mensaje de e-mail. Por ejemplo:

- **DistributionList:** Lista de distribución asociada al proyecto
- **Fecha:** Fecha de recibido el mensaje en la lista de distribución.
- **Subject:** Motivo del mensaje.
- **ConversationTopic:** Tópico de conversación del mensaje
- **SenderName:** usuario que envía el mensaje
- **SenderE-mailAddress:** dirección de correo del usuario que envía el mensaje
- **MensajeTo:** lista de direcciones de correo de los usuarios que reciben el mensaje
- **MensajeCC:** lista de direcciones de correo de los usuarios que reciben el mensaje como copia
- **Tamaño:** tamaño en byte del mensaje
- **Importancia:** propiedad que determina la importancia del mensaje, según el usuario que envía el mensaje
- **Adjuntos:** propiedad que indica si el mensaje tiene documentos adjuntos.

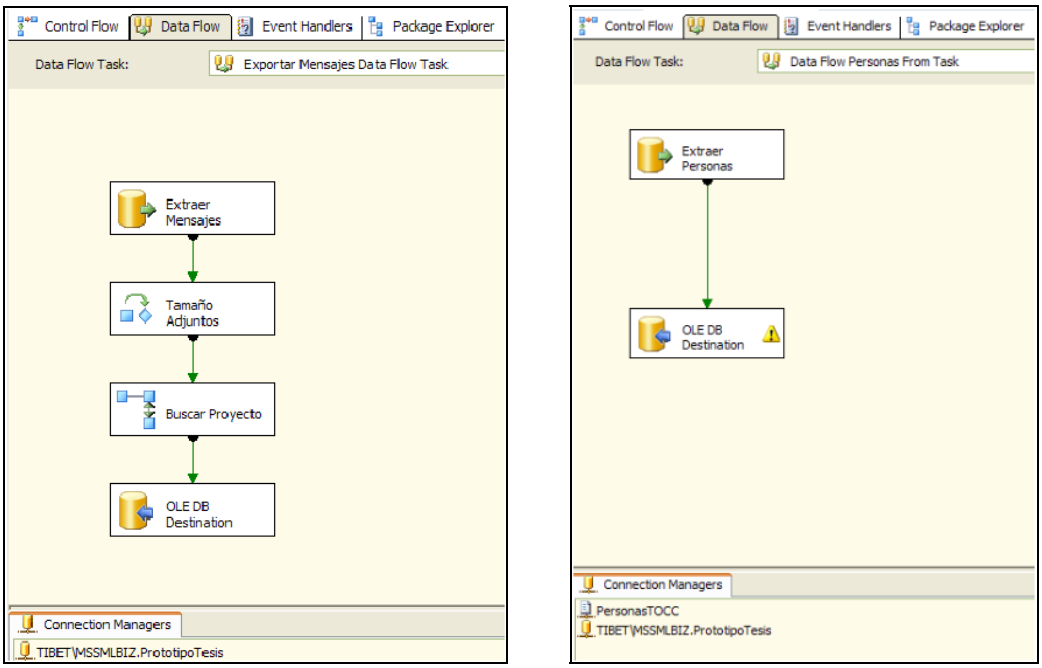


Figura 15 - ETL asociados al prototipo

3.6.7 Vista de Servicios

En ésta vista de presentan los servicios tanto brindados como consumidos por los distintos actores del prototipo.

La siguiente tabla muestra los servicios identificados en el prototipo, quien los provee y quien los consume:

	Provee	Consume
Sistema SPAGPE		• *ExportarMails* • *ExportarMailsporProyecto*
EDM: Agente Extractor de Mensajes	• *ExportarMails* • *ExportarMailsporProyecto*	• *ExportarMails* • *ExportarMailsporProyecto*

Tabla 11 – Servicios identificados para el prototipo propuesto

Contratos

En esta sección se detallan en forma de contratos de servicio, los servicios más relevantes para el prototipo dentro de los previamente listados.

Los contratos tienen una estructura bien definida la cual se especifica a continuación:

Proveedor	Actor responsable de brindar el servicio
Nombre	Nombre identificador del servicio
Responsabilidades	Actividades a ser realizadas
Entrada	Parámetros que el servicio espera recibir
Salida	Parámetros que el servicio retornará
Precondición	Condiciones que se deben cumplir previamente a la invocación del servicio
Poscondición	Condiciones que se dan por cumplidas una vez que el servicio haya culminado
Protocolo	Secuencia en la que distintos servicios de mayor granularidad deben ser invocados para la realización del servicio de menor granularidad

Tabla 12 - Estructura de contratos de servicios

Proveedor: EDM

Nombre: svcExportarMails

Responsabilidades:
Ingresar los e-mails asociados a todas las listas de distribución de los proyectos definidos en SPAGPE.

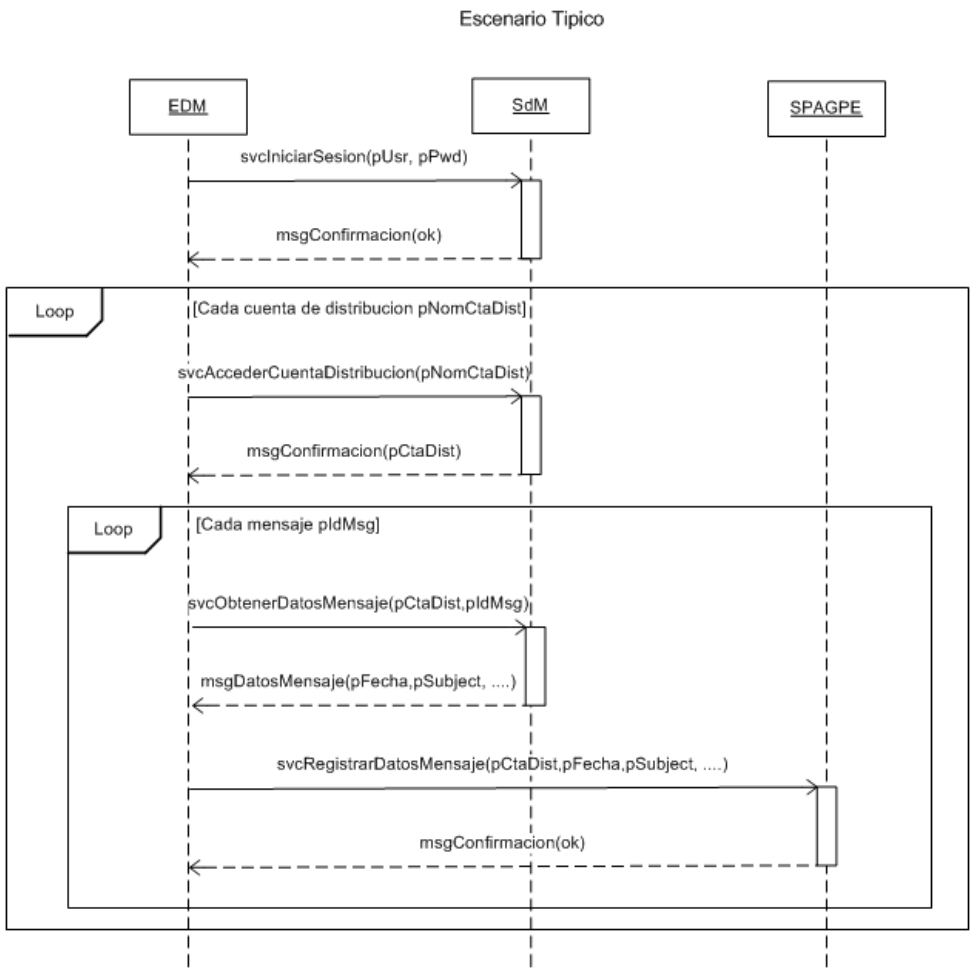
Entrada:
{Nombre de las cuentas de Listas de Distribución}

Salida:
Notificación de si la exportación de mensajes asociados a las cuentas de las listas de distribución se realizó correctamente.

Precondición:
Las cuentas asociadas a las listas de distribución existen en el sistema de e-mail.

Poscondición:
Los mensajes correspondientes a las listas de distribución fueron exportados al prototipo SPAGPE.

Protocolo:



Proveedor: EDM

Nombre: svcExportarMailsporProyecto

Responsabilidades:
Ingresar los e-mails de una cuenta asociada a una lista de distribución de un proyecto definido en SPAGPE.

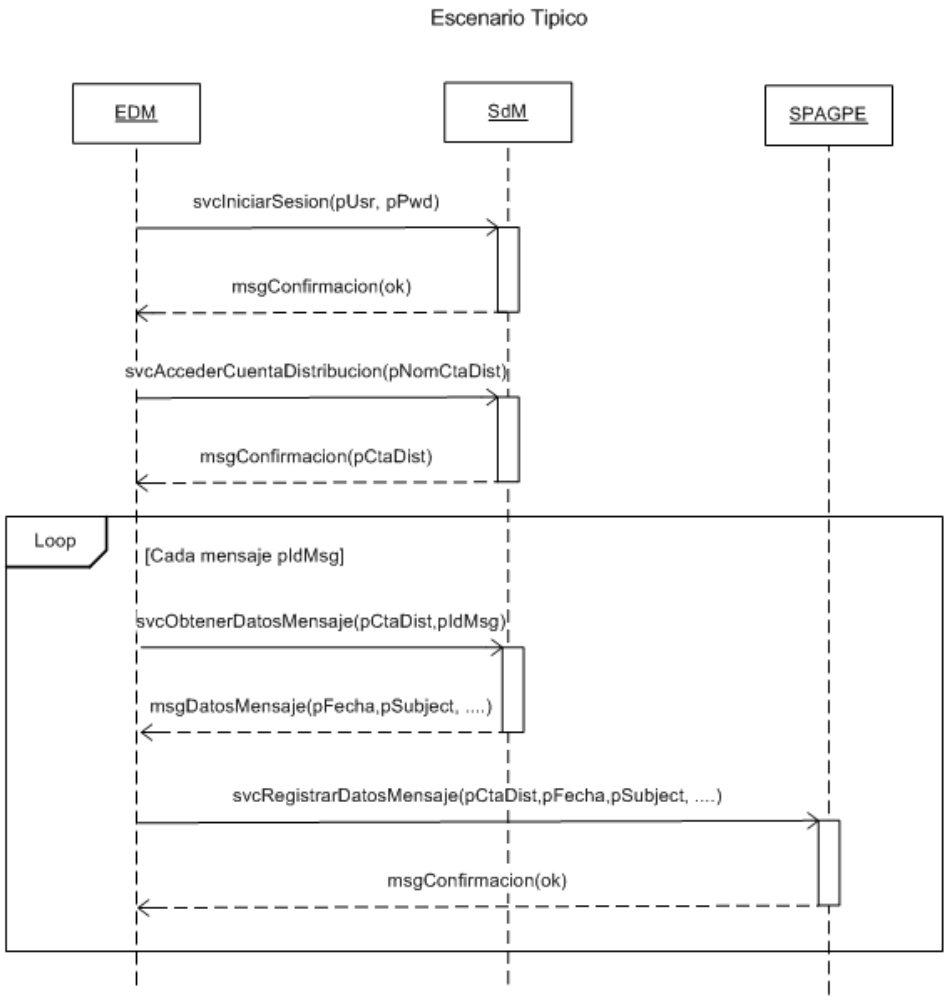
Entrada:
Lista de Distribución

Salida:
Notificación de si la exportación de mensajes asociados a la lista de distribución fue realizada correctamente.

Precondición:
La cuenta asociada a la lista de distribución existe en el sistema de e-mail.

Poscondición:
Los mensajes correspondientes a la cuenta de la lista de distribución fueron exportados al prototipo SPAGPE.

Protocolo:



3.6.8 Vista de Datos

En esta vista se presenta el modelo de datos utilizado en la elaboración del prototipo.

Modelos de Datos

El modelo de datos presenta las relaciones existentes entre las principales entidades analizadas en el caso de estudio Proyecto, Mensaje y RelacionMensaje.

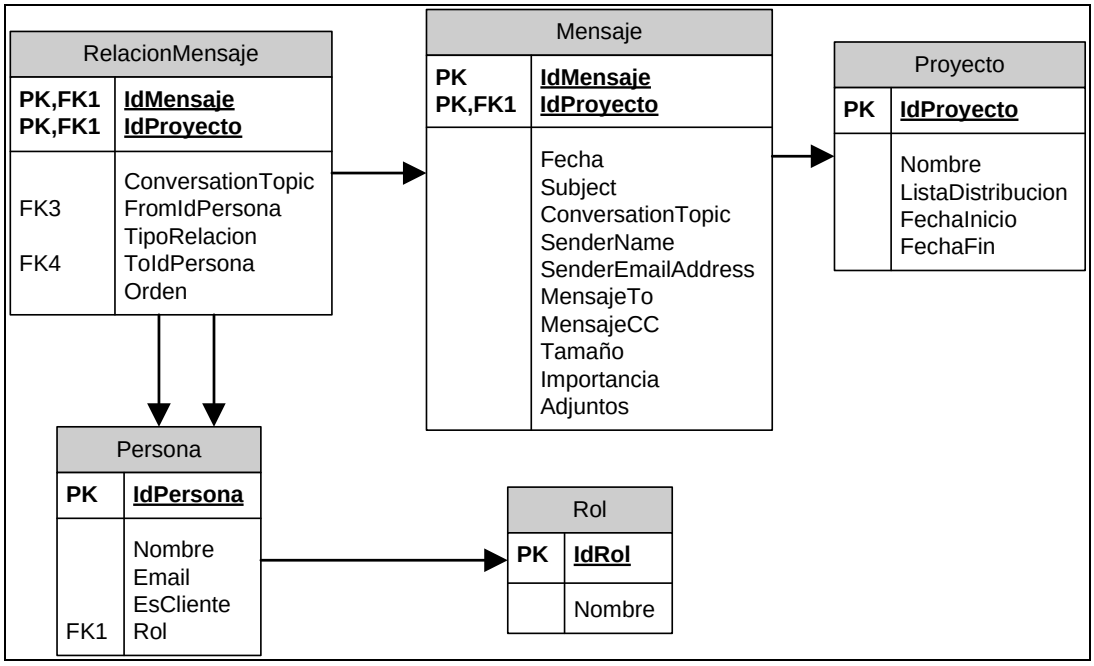


Figura 16 - Modelo de Datos del Prototipo

3.6.9 Vista de Deployment

La vista de deployment presenta la infraestructura necesaria para soportar el prototipo SPAGPE. Se presenta aquí la arquitectura técnica de la aplicación indicando los nodos presentes en la infraestructura tecnológica esperada, y la localización de los componentes en dichos nodos.

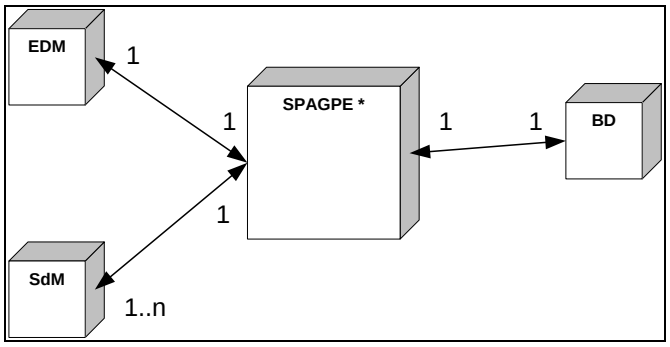


Figura 17 – Vista de deployment

La infraestructura cuenta con 4 nodos básicos. El primero está relacionado con la ubicación de SPAGPE. Este puede tener varias instancias corriendo, una por cada uno

de los nodos de usuario que necesiten hacer uso del prototipo. Otro es el correspondiente al EDM, el cual está asociado a un único nodo, desde el cual se realizan todas las actividades de extracción de mensajes de e-mail. Este nodo está relacionado con el nodo de SPAGPE.

Adicionalmente la infraestructura prevé dos nodos adicionales, el primero en donde se aloja el sistema de e-mail desde donde se realiza la extracción de la información de los mensajes utilizados en el prototipo, y el segundo nodo, correspondiente al de base de datos en donde reside la información utilizada por SPAGPE.

3.7 Resultado Obtenido

Con el proceso definido para la resolución del caso de estudio se dan respuestas a varias de las preguntas formuladas al inicio de la sección y cuyo resultado sirve como apoyo a la detección de patrones y antipatrones de gestión.

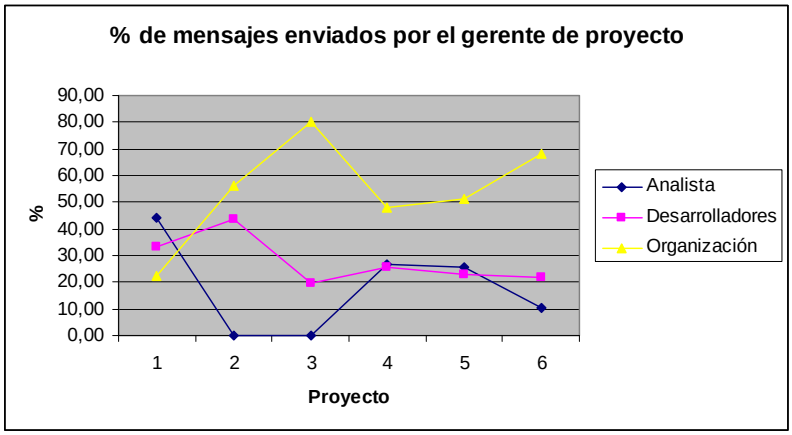
De los 17 proyectos relevados, se detectó que en 6 de ellos la cantidad de mensajes en las comunicaciones entre el gerente de proyecto y el equipo de desarrolladores era mayor que la comunicación entre el gerente y el analista, o cercana a esta. En la tabla 13, se observa el porcentaje de mensajes enviados por el gerente de proyectos en forma interna para estos proyectos.

Proyecto	Analista	Desarrolladores	Organización	Total
1	44,44%	33,33%	22,22%	100%
2	0,00%	43,75%	56,25%	100%
3	0,00%	19,67%	80,33%	100%
4	26,67%	25,56%	47,78%	100%
5	25,76%	22,73%	51,52%	100%
6	10,29%	21,57%	68,14%	100%

Tabla 13 - % de mensajes enviados por el gerente de proyecto en forma interna.

Según se analizó en 3.5.3, esta relación de las comunicaciones entre el gerente y equipo de proyecto es uno de los síntomas del antipatrón *Micromanagement*. Se genera una fuerza desbalanceada en la gestión de los recursos a nivel de comunicaciones que puede ser la causa de la pérdida de foco y de un control desbalanceado por parte del gerente en los demás aspectos del proyecto, como son la tecnología y los procesos. La falla en estos puntos puede derivar en consecuencias como el desarrollo de un producto equivocado y con un sobrecosto no planificado, en fallas tecnológicas, en el desgaste del equipo y la finalización prematura del proyecto.

Observando la gráfica correspondiente a los datos de la tabla 13, se nota que existe una mayor comunicación de parte del gerente con el resto de la organización (roles de servicios). También se observa que la comunicación en forma cuantitativa del gerente hacia el equipo de desarrollo es mayor o cercana a su comunicación con el analista del equipo.



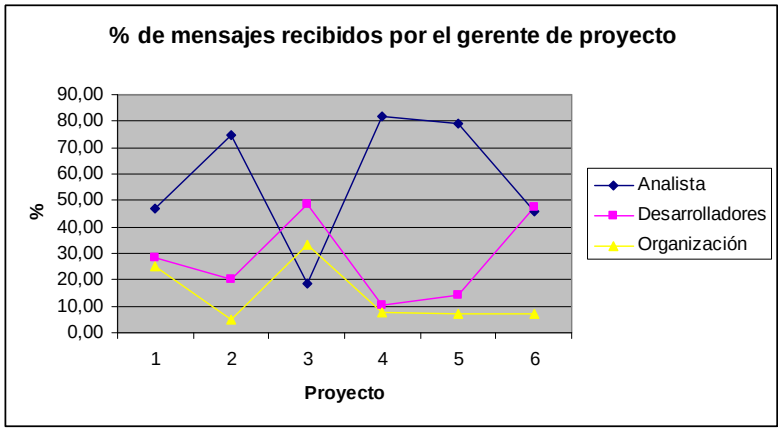
Para los mismos proyectos se analizan el % de mensajes recibidos por el gerente de proyecto desde el equipo de proyecto y en el mismo periodo. Se observa que para 2 de ellos el % de mensajes enviados por el equipo de desarrolladores es mayor al % de mensajes enviados por el analista al gerente del proyecto.

Proyecto	Analista	Desarrolladores	Organización	Total
1	46,88%	28,13%	25,00%	100%
2	74,58%	20,34%	5,08%	100%
3	18,33%	**48,33%**	33,33%	100%
4	82,09%	10,45%	7,46%	100%
5	79,07%	13,95%	6,98%	100%
6	45,61%	**47,37%**	7,02%	100%

Tabla 14 - % de mensajes recibido por el gerente de proyecto en forma interna.

En particular para los proyectos 2 y 3, la vía de comunicación desde el gerente al analista es nula, ver tabla 13, pero se observa que la comunicación desde el analista hacia el gerente de proyecto es de un 74,58% en el primer caso y de un 18,33% en el segundo.

Esta diferencia puede estar indicando una falla de comunicación entre el gerente y el analista del proyecto, en donde el analista no tiene una comunicación fluida y una respuesta a sus mensajes por parte del gerente. En estos casos se puede estar adicionalmente ante la presencia de síntomas del antipatrón *Irrational Management* (2.6.4) y puede servir para realizar un análisis con más detalle para entender este comportamiento de comunicación que se presenta.



En el resto de los proyectos analizados no presentan síntomas cuantitativos de estar ante la presencia del antipatrón Micromanagement, Tabla 15.

Proyecto	Analista	Desarrolladores	Organización	Total
7	66,67%	11,11%	22,22%	100%
8	62,26%	1,89%	35,85%	100%
9	57,14%	0,00%	42,86%	100%
10	36,36%	24,24%	39,39%	100%
11	80,00%	0,00%	20,00%	100%
12	52,33%	16,28%	31,40%	100%
13	46,85%	13,51%	39,64%	100%
14	21,05%	11,58%	67,37%	100%
15	63,16%	21,05%	15,79%	100%
16	17,65%	11,76%	70,59%	100%
17	100,00%	0,00%	0,00%	100%

Tabla 15 - % de mensajes enviados por el gerente de proyecto en forma interna.

Para la mayoría de los proyectos, la comunicación entre el gerente de proyecto y el analista en términos de % de mensajes enviados y recibidos (Tabla 15 y Tabla 16), no presentan diferencias excepcionales que hagan pensar que se están ante la presencia del antipatrón *Irrational Managament*. Particularmente se pueden estudiar y profundizar aquellos que tienen una diferencia mayor a un 40%, como es el caso de los proyectos 9, 13 y 16 que presentan diferencias de un 40,36%, 45,61% y un 61,35% respectivamente. En estos casos la diferencia es en la comunicación desde el analista hacia el gerente de proyecto.

Proyecto	Analista	Desarrolladores	Organización	Total
7	84,48%	3,45%	12,07%	100%
8	92,11%	0,00%	7,89%	100%
9	97,50%	0,00%	2,50%	100%
10	20,00%	0,00%	80,00%	100%
11	100,00%	0,00%	0,00%	100%
12	72,22%	12,50%	15,28%	100%
13	92,46%	0,00%	7,54%	100%
14	50,00%	23,08%	26,92%	100%
15	81,94%	9,72%	8,33%	100%
16	79,00%	3,00%	18,00%	100%
17	93,65%	6,35%	0,00%	100%

Tabla 16 - % de mensajes recibido por el gerente de proyecto en forma interna.

El análisis consolidado de todos los proyectos nos puede indicar el comportamiento de la organización al nivel del porfolio de proyectos y determinar la existencia de patrones y antipatrones no solo al nivel de un proyecto sino también a nivel de la organización.

4 Conclusiones y Trabajos futuros

"Lo importante es no dejar de hacerse preguntas."
Albert Einstein
(1879 - 1955)

Se presentan las conclusiones del informe en relación al estudio de patrones y antipatrones de gestión de proyectos de software, y las expectativas que se tienen de cara a los trabajos futuros que se den dentro del área.

4.1 Conclusiones y Trabajos futuros

El informe comprende en una primera instancia el estudio del estado de arte en la gestión de proyectos. Este estudio está particularmente enfocado al área de patrones y antipatrones de gestión de proyectos de software, en donde se trata la situación actual de la gestión de proyectos, se introducen los conceptos de gestión de portafolios y de programas, se relevan los modelos de madurez utilizados para la ejecución de este tipo de proyectos, así como también sus diferentes áreas de aplicación.

Los aspectos de la situación actual de la gestión de proyectos son presentados en forma breve, repasando los conceptos básicos y diferencias existentes entre la gestión de portafolios de proyectos y gestión de programas. El gerente de proyectos debe comprender que estos conceptos y sus diferencias, le ayudarán a reconocer los contextos en donde se ejecutan sus proyectos, y le servirán a entender el propósito de cada uno de estos y a poder enmarcarlos dentro del contexto más adecuado, poniendo foco y haciendo énfasis en la planificación a mediano o largo plazo que esté alineado según la planificación asociada a la gestión del portafolio y del propio programa. En caso de existir este tipo de gestión en la organización, la gestión del proyecto debe estar en sintonía. En caso de no existir, el gerente puede ser capaz de ayudar a introducir estos conceptos en su organización.

Durante el relevamiento realizado no se encontraron patrones y antipatrones propios de la gestión de portafolios y programas. Ante esto, el presente informe puede ser un punto de partida para trabajos futuros, el cual puede ser complementado con un estudio y relevamiento en profundidad de los aspectos a ser tenidos en cuenta en este tipo de gestiones. Para una introducción al tema se puede comenzar consultando en [ROSS07], [AMBLER08], [ASH04].

El informe presenta y hace referencia al concepto relacionado a los modelos de madurez actuales de la gestión de proyectos, y que son definidos por las organizaciones reconocidas en la industria actual. Estos modelos se basan en objetivos comunes [2.1.2].
La mayoría de los gerentes de proyectos locales y de la región que están asociados a algunas de estas organizaciones, principalmente al PMI [PMICM]. En el caso del PMI y durante el relevamiento realizado se encontró como única referencia directa a patrones y antipatrones de proyectos, el trabajo realizado por Deepak [PP06], relacionados a los patrones de gestión efectiva en proyectos distribuidos. De los 60 trabajos de investigación presentados en el 2008, 6 fueron aprobados para ser ejecutados en el 2009. Estos trabajos tienen como objetivo relacionado estudiar algún tipo de patrón relacionado con la gestión de proyectos, lo que está implicando que este tipo de estudio está teniendo interés dentro de la comunidad, lo que seguramente tendrá un impacto positivo dentro del área de la gestión.
Por ejemplo:

- En "Early Warning Signs in Complex Projects" se plantea la investigación de la práctica de evaluación de proyectos, de forma de identificar como y en qué grado las primeras señales de advertencia de problemas, fallas, bajo rendimiento y un desvío en el costo de un proyecto se pueden llegar a identificar. Esta investigación tiene puntos de contacto con la detección de síntomas, que es uno

de los puntos que definen a los antipatrones. Se estima que el trabajo esté finalizado en último cuatrimestre del 2009.

- En "Project Management and Organizational Change", a finalizar en el tercer cuatrimestre del 2010, los investigadores se plantean realizar un estudio de investigación para identificar la naturaleza de la organización y los cambios de comportamiento en la gestión de proyectos, y como estos varían según el tipo de proyecto, según el contexto, el grado de organización y el cambio de comportamiento que esté involucrado. Adicionalmente el estudio se plantea explorar los patrones de involucramiento de las personas con experiencia en desarrollo organizacional y gestión de proyectos, temas que tiene una introducción en el trabajo realizado por Laplante relacionados a los antipatrones que están orientados al ambiente y cultura de la organización.

- "Best Industry Outcomes", investigación a finalizar en el último cuarto del 2010, tiene como objetivo identificar los patrones de gestión de proyectos asociados a los tipos de resultados que suelen ser más valorados por las organizaciones.

También se presentaron los modelos de ciclos de vida de proyectos generalmente utilizados en la industria del software, que son los escenarios en donde se suceden los patrones y antipatrones estudiados en el informe, los cuales pueden llegar a variar según el modelo utilizado sea predictivo o ágil.

El estado del arte incluyó como parte central el relevamiento de patrones y antipatrones, sus diferentes clasificaciones, así como también la forma de identificarlos dentro de los contextos y escenarios usuales.

Se presentaron los modelos de referencias de patrones y antipatrones, particularmente los propuestos por Brown. La bibliografía estudiada en relación con el tema toma como referencias y como una base de partida para la elaboración y definición de patrones y antipatrones lo que es descrito por Brown et al; como es el caso del estudio realizado Laplante y Neill. La enumeración de las causas raíces que Brown propone como las causas que lleva a la falla de proyectos, son aspectos que no se encontraron como parte del relevamiento de las metodologías o modelos de madurez mencionados en el punto 2.1.2.

Los modelos mencionados generalmente no consideran la influencia que tiene el comportamiento humano sobre el software, y tampoco las variables culturales de cada organización en donde se vaya a aplicar alguno de estos modelos. Brown incluye acertadamente a las causas raíces basadas en el comportamiento humano y a las fuerzas principales que actúan a la hora de tomar decisiones, y Laplante las extiende acertadamente a los comportamientos dentro de las organizaciones donde transcurren los proyectos.

En las secciones 2.4.1 y 2.4.2 se presentó según la propuesta de Brown una clasificación de los patrones y antipatrones, basada en las personas, los procesos y la tecnología. Esta clasificación puede ser enriquecida con los patrones de gestión en metodologías ágiles y de gestión de proyectos distribuidos, que generan otro tipo de patrones y antipatrones los cuales pueden ser estudiados y profundizados en [CBURN05], [DEAN07], [DEAN09], [APAT], [DEEPAK07], [VAL08]. Se presentan aquellos que a mi criterio e interés encuentro elementales para la generación de un lenguaje de patrones que esté

orientado a la gestión de proyectos, y que están relacionados e involucran directamente al rol de gerente de proyecto, y cuyo resultado permite ayudar a entender y a comprender los conceptos asociados a los patrones y antipatrones de gestión de proyectos de software, principalmente desde el punto de vista del gerente de software, introduciendo el área de investigación y su aplicación a los proyectos de software. La enumeración de los patrones y antipatrones es más extensa que la presentada en el informe y como plantean los autores de gran parte de ellos, la creación de uno nuevo se debe basar en el resultado de varias experiencias prácticas de forma que sirvan para ayudar a una mejor compresión de las causas y consecuencias a situaciones generadas dentro del contexto de la ejecución de un proyecto de software.

El informe presenta la forma de cómo describirlos y propone una alternativa común para estandarizar los modelos propuestos por dos de los autores mencionados y referenciados, Brown y Laplante. Existe un interesante área de desarrollo, desde donde se pueden aportar herramientas que sigan la línea anterior planteada y que ayuden a los gerentes de proyecto o responsables de liderarlos a identificar, prever y recuperarse de situaciones que están representadas por el conjunto de patrones y antipatrones enumerados en el informe.

Los patrones y antipatrones deben ser tenidos en cuenta por las organizaciones de desarrollo de software, y deben ser considerados como una herramienta más de apoyo a la gestión y decisiones que se den dentro de un proyecto de software. Para esto se debe contar con herramientas que ayuden a enriquecer el conjunto de patrones y antipatrones, manteniendo actualizado dicho conjunto en base a las diferentes experiencias y lecciones aprendidas que se dan en las situaciones reales surgidas de sus aplicaciones.

Durante y tras el estudio del arte realizado en el informe, no se encontraron herramientas de este tipo que apoyen al descubrimiento específico de patrones y antipatrones de gestión de proyectos. Esto particularmente motivó para una segunda instancia del informe la implementación de un prototipo que sirva como base para el análisis y descubrimiento de patrones y antipatrones de gestión y la posterior aplicación a un caso de estudio real para dos de los antipatrones enumerados en el informe: *Micromanagement* e *Irrational Management*.

Como trabajo futuro se plantea que la implementación de este tipo de aplicaciones puedan seguir la línea del prototipo planteado, basándose en soluciones que apliquen técnicas de data warehouse y data mining. Es deseable la integración con los sistemas de Recursos Humanos como base de datos de los integrantes de la organización y de los proyectos en particular, desde donde se puede realizar un mapeo con la información contenida dentro del sistema, como ser datos personales, fecha de ingreso, historia laboral dentro de la empresa, capacitación y también sistemas de desarrollo profesional o de gestión de rendimiento individual. Las herramientas de gestión de patrones y antipatrones se pueden asociar a los sistemas de gestión de proyectos, en donde se definen sus costos y su planificación de tiempo. Esta integración puede ayudar a responder preguntas asociadas a los niveles de comunicaciones existentes en las diferentes etapas del ciclo de vida de un proyecto de software, definiendo los patrones para que este sea exitoso, teniendo como variables adicionales y de referencia, el presupuesto asignado, costo planificado y actual del proyecto y la satisfacción del cliente.

Ante el equipo de desarrollo, sería deseable la integración a sistemas de repositorios de código, a partir del cual se puedan estudiar los patrones y antipatrones de gestión de la

configuración, especialmente a través de las actualizaciones de las diferentes piezas de código que forman parte de una solución, y con esto determinar el grado de actividad de los equipos de trabajo.
Todos los puntos de integración anteriores sirven como apoyo al descubrimiento de patrones y antipatrones. Tener y compartir catálogos de patrones y antipatrones a través de mecanismos de comunicación ayuda a generar y construir sistemas de redes sociales a nivel de empresas y equipos de proyecto, siendo este un gran desafío como objetivo para trabajos futuros.

5 Anexos

"La brevedad es el alma del ingenio."
William Shakespeare
(1564 - 1616)

Este anexo contiene información asociada a la implementación del prototipo. Se muestra alguno de los algoritmos utilizados en la resolución al problema planteado en el capitulo correspondiente al caso de estudio.

5.1 Anexo

Este anexo contiene información asociada a la implementación del prototipo. Se muestran alguno de los algoritmos utilizados en la resolución al problema planteado.

```
-- ============================================================================
-- Author:            Leonardo Dominguez
-- Description:       Extracción de mensajes y carga en base de datos.
-- ============================================================================
Option Strict Off
Option Explicit On
Imports System.Data
Imports System.Data.SqlClient

Friend Class FrmExtraerEmails
Inherits System.Windows.Forms.Form
Const FOLDER_TO_OPEN As String = "FOLDER_TO_OPEN"
Const  CONNECTIONSTRING  As  String  =  "server=DES-leonardod;database=PrototipoTesis;
uid=; pwd=;"


Private Sub FrmExtraerEmails_Load(ByVal eventSender As System.Object, ByVal eventArgs
As System.EventArgs) Handles MyBase.Load
       Dim olMAPI As Microsoft.Office.Interop.Outlook.NameSpace
        Dim oFolder As Microsoft.Office.Interop.Outlook.MAPIFolder
        Dim oFolder1 As Microsoft.Office.Interop.Outlook.MAPIFolder
        Dim oFolder2 As Microsoft.Office.Interop.Outlook.MAPIFolder
        Dim oFolder3 As Microsoft.Office.Interop.Outlook.MAPIFolder


       olMAPI  =  CreateObject("Outlook.Application",  "localhost").GetNamespace("MAPI")

        oFolder1 = olMAPI.Folders("Public Folders")
        oFolder2 = oFolder1.Folders("Favorites")
        oFolder3 = oFolder2.Folders("FolderName")

        Call GetFolderEmailMessages(oFolder3, "->")


        GC.Collect()
        GC.WaitForPendingFinalizers()
        GC.Collect()
        GC.WaitForPendingFinalizers()

        olMAPI = Nothing

       End Sub


Sub GetFolderEmailMessages(ByRef tempfolder As
Microsoft.Office.Interop.Outlook.MAPIFolder, ByRef a As String)
       Dim i As Short
        Dim oMessage As Object
        Dim sSql As String
        Dim sTo As String
        Dim sCC As String


       For Each oMessage In tempfolder.Items
          If TypeName(oMessage) = "MailItem" Then
            If oMessage.To.ToString.Length <= 1000 Then
                sTo = oMessage.To.ToString.Replace("'", "")
            Else
                sTo = oMessage.To.ToString.Replace("'", "").Substring(1, 1000)
            End If

            If oMessage.CC.ToString.Length <= 1000 Then
                sCC = oMessage.CC.ToString.Replace("'", "")
```

```
            Else
                sCC = oMessage.CC.ToString.Replace("'", "").Substring(1, 1000)
            End If

                sSql = "INSERT INTO MensajesExportados([DistributionList],[Fecha],
                [Subject],[ConversationTopic],[SenderName]," & _
                "[SenderEmailAddress],[MensajeTo],[MensajeCC],
                [Tamaño],[Importancia],[Adjuntos]) " & _
                "VALUES ('" & FOLDER_TO_OPEN & "', '" & oMessage.ReceivedTime & "', '"
                & _ oMessage.Subject.ToString.Replace("'", " ") & "', '" & _
                oMessage.ConversationTopic.ToString.Replace("'", " ") & "', '" &
                oMessage.SenderName & "','" & _
                oMessage.SenderEmailAddress & "','" & sTo & "','" & sCC & "', '" &
                oMessage.Size & "', '" & _
                oMessage.Importance & "', '" &      oMessage.Attachments.count & "' )"
               Try

                      CreateCommand(sSql, CONNECTIONSTRING)

               Catch ex As Exception
               End Try
          End If
         Next oMessage
        End Sub

Public Sub CreateCommand(ByVal queryString As String, ByVal connectionString As
String)
         Using connection As New SqlConnection(connectionString)
         Dim command As New SqlCommand(queryString, connection)

            command.Connection.Open()
            command.ExecuteNonQuery()
            command.Connection.Close()
         End Using
        End Sub

End Class


-- ================================================================
-- Author:            Leonardo Dominguez
-- Description:       Carga de la tabla Persona en base a la
-- información recolectada de los mensajes.
-- ================================================================
ALTER PROCEDURE [dbo].[AgregarPersona]
        -- Add the parameters for the stored procedure here
        @NombreE-mail  nchar(255),
        @IdPersona            int = 0 out

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;


/*
a- Si la misma existe en Persona.Nombre o Persona.E-mail, no hacer nada.
b- Si es formato E-mail => buscar en Persona.E-mail
        Si no existe en Persona.E-mail
        => cargar en Persona.E-mail
        => cargar cuenta en persona (parte izquierda de cuenta@dominio)
c- Si  no es formato e-mail => buscar en persona.Nombre
        Si no existe en Persona.Nombre
        => cargar en Persona.Nombre.

*/
        Declare @CuentaE-mail nchar(255)

        Set @CuentaE-mail = dbo.TransformarCuentaE-mail(@NombreE-mail)

        -- Tomamos el primero de los registros, asumimos que mas de 2 registros
        pertenecen a la misma persona
        -- Utilizar en el algoritmo de generacion de RelacionMensaje con la persona que
        envia el mensaje
        Set @IdPersona  = dbo.ObtenerIdPersona(@NombreE-mail, @CuentaE-mail)
```

```
        if (@IdPersona is null)
                If PatIndex('%@%',@CuentaE-mail) <> 0  -- Tiene formato E-mail
                        Begin
                                Insert into Persona(Nombre, E-mail, EsCliente, Rol)
                                Values (Left(@CuentaE-mail, PatIndex('%@%',@CuentaE-
                                mail)-1), @CuentaE-mail, null, null)
                                Set @IdPersona = @@identity
                        end

                Else
                        Begin
                                Insert into Persona(Nombre, E-mail, EsCliente, Rol)
                                Values (@NombreE-mail, null, null, null)
                                Set @IdPersona = @@identity
                        End

END


-- =============================================================
-- Author:            Leonardo Dominguez
-- Description:       Carga de la tabla RelacionMensaje.
-- =============================================================
ALTER PROCEDURE [dbo].[AgregarRelacionMensaje]
        -- Add the parameters for the stored procedure here
        @IdMensaje bigint,
        @ConversationTopic nchar(500),
        @FromIdPersona int,
        @TipoRelacion char(1),
        @ToIdPersona int,
        @Orden int
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;


    -- Insert statements for procedure here
        Insert into RelacionMensaje(IdMensaje, ConversationTopic, FromIdPersona,
        TipoRelacion, ToIdPersona, Orden)
        Values (@IdMensaje, @ConversationTopic, @FromIdPersona, @TipoRelacion,
        @ToIdPersona, @Orden)

END


-- =============================================================
-- Author:            Leonardo Dominguez
-- Description:       Carga de la tabla de Mensajes exportados desde SdM
-- =============================================================
ALTER PROCEDURE [dbo].[ExportarMensaje]
(
    @DistributionList BigInt,
    @Fecha nchar(10),
    @Subject nchar(10),
    @ConversationTopic nchar(10),
        @SenderName nchar(10),
        @SenderE-mailAddress nchar(10),
        @MensajeTO nchar(10),
        @MensajeCC nchar(10),
        @Tamaño nchar(10),
        @Importancia nchar(10),
        @Adjuntos nchar(10)
)
AS
DECLARE @ReturnValue INT
SELECT @ReturnValue = 1


INSERT INTO MensajesExportados
(
    DistributionList,
    Fecha,
    Subject,
    ConversationTopic,
        SenderName,
```

```
        SenderE-mailAddress,
        MensajeTO,
        MensajeCC,
        Tamaño,
        Importancia,
        Adjuntos
)
VALUES
    (@DistributionList,
    @Fecha,
    @Subject,
    @ConversationTopic,
        @SenderName,
        @SenderE-mailAddress,
        @MensajeTO,
        @MensajeCC,
        @Tamaño,
        @Importancia,
        @Adjuntos)

RETURN @ReturnValue


-- ================================================================
-- Author:            Leonardo Dominguez
-- Description:       Dado un nombre o e-mail, obtiene el IDPersona de
-- la tabla Persona
-- ================================================================
ALTER FUNCTION [dbo].[ObtenerIdPersona]
(
        -- Add the parameters for the function here
        @Nombre nchar(255),
        @E-mail nchar(100)
)
RETURNS int
AS
BEGIN
        -- Declare the return variable here
        DECLARE @IdPersona int

        Set @IdPersona  = 0
        Set @IdPersona  = (Select Top 1 IdPersona
                                        From Persona
                                        Where E-mail = @E-mail)

        if (@IdPersona is null)

                Set @IdPersona  = (Select Top 1 IdPersona
                                                        From Persona
                                                        Where Nombre = @Nombre)



        -- Return the result of the function
        RETURN @IdPersona

END
```

```
-- =============================================================
-- Author:            Leonardo Dominguez
-- Description:       Funcion que recibe el dato CuentaE-mail obtenido
-- desde el header del mensaje y lo transforma en una cuenta de e-mail
-- con formato nombre@dominio.com
-- =============================================================
ALTER FUNCTION [dbo].[TransformarCuentaE-mail]
(
       -- Add the parameters for the function here
       @CuentaE-mail nchar(255)
)
RETURNS nchar(255)
AS
BEGIN
       -- Declare the return variable here
       DECLARE @Result nchar(255)

       -- Add the T-SQL statements to compute the return value here
       SELECT @Result =
              CASE
                     WHEN (SUBSTRING(@CuentaE-mail, 1, 11) = '/O=COMPANY') THEN
                     rtrim(substring(@CuentaE-mail,PATINDEX('%/CN=RECIPIENTS/CN=%',
                     @CuentaE-mail)+18,len(@CuentaE-mail))) + '@Company.com.uy'
                     ELSE @CuentaE-mail
              END

       -- Standarizamos los nombres y cuentas
       -- Nombres entre "", (), con nomenclatura ldap. en formato cuenta@dominion,
       -- Nombre y Apellido.
       Set @Result = ltrim(rtrim(replace(@Result,'"','')))
       Set @Result = ltrim(rtrim(replace(@Result,'(','')))
       Set @Result = ltrim(rtrim(replace(@Result,')','')))
       Set @Result = ltrim(rtrim(replace(@Result,'<','')))
       Set @Result = ltrim(rtrim(replace(@Result,'>','')))

       -- Return the result of the function
       RETURN @Result

END


-- =============================================================
-- Author:            Leonardo Dominguez
-- Description:       Consultas SQL para obtener informacion relacionada
-- a los mensajes extraidos del SdM
-- =============================================================

-- Cantidad de Proyectos
Select count(*) from proyecto


-- #Cantidad de Mensaje por Proyecto
Select P.Nombre, count(*) as 'Cantidad de Mensajes'
from Mensaje M Inner Join Proyecto P on M.IdProyecto = P.IdProyecto
where (M.Fecha >= '01/01/08' and M.Fecha <= '06/30/08')
group by P.Nombre

-- # de Personas en las comunicaciones
Select P.Nombre, count(*) as 'Cantidad de Mensajes'
from Mensaje M Inner Join Proyecto P on M.IdProyecto = P.IdProyecto
where (M.Fecha >= '01/01/08' and M.Fecha <= '06/30/08')
group by P.Nombre

Select  distinct P. Nombre
from Mensaje M inner join RelacionMensaje RM on M.IdMensaje = RM.IdMensaje
inner join Persona P on RM.FromIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and M.Fecha <= '06/30/08')
Union
Select  distinct P. Nombre
from Mensaje M inner join RelacionMensaje RM on M.IdMensaje = RM.IdMensaje
inner join Persona P on RM.ToIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and M.Fecha <= '06/30/08')
```

```
-- # de Personas participantes en el equipo de proyecto
Select  PR.IdProyecto, PR.Nombre, count(*) as CantPersonasCliente
from Proyecto PR inner join Mensaje M on PR.IdProyecto = M.IdProyecto
inner join RelacionMensaje RM on M.IdMensaje = RM.IdMensaje
inner join Persona P on RM.FromIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and Mb.Fecha <= '06/30/08') and P.EsCliente = 0
Group by PR.IdProyecto, PR.Nombre

-- # de Personas pertenecientes al cliente
-- Contenidas en el From de los mensajes.
-- Participantes activos de la comunicacion
Select  PR.IdProyecto, PR.Nombre, count(*) as CantPersonasCliente
from Proyecto PR inner join Mensaje M on PR.IdProyecto = M.IdProyecto
inner join RelacionMensaje RM on M.IdMensaje = RM.IdMensaje
inner join Persona P on RM.FromIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and M.Fecha <= '06/30/08') and P.EsCliente = 1
Group by PR.IdProyecto, PR.Nombre

-- Contenidas en el From de los mensajes.
-- Participantes pasivos de la comunicacion
Select  PR.IdProyecto, PR.Nombre, count(*) as CantPersonasCliente
from Proyecto PR inner join Mensaje M on PR.IdProyecto = M.IdProyecto
inner join RelacionMensaje RM on M.IdMensaje = RM.IdMensaje
inner join Persona P on RM.FromIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and M.Fecha <= '06/30/08') and P.EsCliente = 1
Group by PR.IdProyecto, PR.Nombre


-- # de Personas participantes en el proyecto
Select M.IdProyecto, RM.FromIDPersona, P.Nombre, P.Escliente
From Mensaje M
inner join RelacionMensaje RM on  RM.IdMensaje = M.IdMensaje
inner join Persona P on RM.FromIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and M.Fecha < '07/01/08')
union
Select M.IdProyecto, RM.TOIDPersona, P.Nombre, P.Escliente
From Mensaje M
inner join RelacionMensaje RM on  RM.IdMensaje = M.IdMensaje
inner join Persona P on RM.ToIdPersona = P.IdPersona
Where (M.Fecha >= '01/01/08' and M.Fecha < '07/01/08')
Order by M.IdProyecto, P.Escliente


-- Mensajes enviados por el Gerente de Proyecto a los clientes (#6171)
-- considerando TO y CC, sin discriminar al cliente y al equipo en la comunicacion
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
Where RM.FromIdPersona = 138 and P.Escliente = 1

-- Mensajes enviados por el Gerente de Proyecto a los clientes (#2381)
-- considerando solo TO,  sin discriminar al cliente y al equipo en la comunicacion
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
Where RM.FromIdPersona = 138 and P.Escliente = 1 and RM.TipoRelacion = 'T'

-- Mensajes enviados por el Gerente de Proyecto a los clientes (#3790)
-- considerando solo CC,  sin discriminar al cliente y al equipo en la comunicacion
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
Where RM.FromIdPersona = 138 and P.Escliente = 1 and RM.TipoRelacion = 'C'

-- Mensajes enviados por el Gerente de Proyecto al equipo (#4061)
-- considerando TO y CC, sin discriminar al cliente y al equipo en la comunicacion
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
Where RM.FromIdPersona = 138 and P.Escliente = 0

-- Mensajes enviados por el Gerente de Proyecto al equipo (#1494)
-- considerando solo TO,  sin discriminar al cliente y al equipo en la comunicacion
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
Where RM.FromIdPersona = 138 and P.Escliente = 0 and RM.TipoRelacion = 'T'

-- Mensajes enviados por el Gerente de Proyecto al equipo (#2567)
-- considerando solo CC,  sin discriminar al cliente y al equipo en la comunicacion
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
```

```
Where RM.FromIdPersona = 138 and P.Escliente = 0 and RM.TipoRelacion = 'C'

-- Mensajes enviados por el Gerente de Proyecto al equipo (#402)
-- considerando solo T,  sin discriminar al cliente y al equipo en la comunicacion
-- Pero con orden = 1,  A = 297, D = 105, con orden <> 1: A =67  , D= 100
Select count(*)
From RelacionMensaje RM inner join Persona P on RM.ToIdPersona = P.IdPersona
Where RM.FromIdPersona = 138 and RM.TipoRelacion = 'T' and RM.Orden = 1 and
P.Escliente = 0 and P.Rol = 'D'
```

6 Referencias Bibliográficas

[CLE06] David I. Cleland, Roland Gareis. *Global project management handbook*. McGraw-Hill Professional, 2006. ISBN 0071460454.

[LEWIS06] Lewis R. Ireland. *Project Management*. McGraw-Hill Professional, 2006. ISBN 007147160X.

[BMMM98] Brown, W., Malveau, R., Mc Cormick III, H., Mowbray, T. *Antipatterns: Refactoring Software, Architectures and Project in Crisis*. Wiley and Sons, 1998. ISBN 0471197130.

[BMMM00] William J. Brown, Hays W. "Skip" McCormick III, Scott W. Thomas. *AntiPatterns in Project Management*. Wiley and Sons, 2000. ISBN 0471363669.

[BOEHM86] Barry Boehm. *A Spiral Model of Software Development and Enhancement*. ACM SIGSOFT Software Engineering Notes, August 1986.

[ALE77] Christopher Alexander. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977. ISBN 0195019199.

[ALE79] Christopher Alexander. *The Timeless Way of Building*. Oxford University Press, 1979. ISBN 0195024028.

[PMBOK04] *Guía de los fundamentos de la Dirección de Proyectos*. Project Management Institute; 3 edition, 2005. ISBN 1930699735.

[GOF95] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, 1994. ISBN 0201633612.

[FERN99] Edward Fern. *Time-to-Profit Project Management*. Time to Profit Inc. 1999

[FOLG05] Antonio Folgueras et al. *The Art of Project Management: Key adjustments Factors using Dynamic Techniques*. Universidad Carlos III, España, 2005.

[FOWLER96] Martin Fowler *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1997. ISBN 0201895420.

[FOWLER02] Martin Fowler et al. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002. ISBN 0321127420.

[FRAME05] J.Davidson Frame. *La nueva dirección de proyectos*. Granica, 2005. ISBN 9506411271.

[LAP06] Phillip Laplante, Colin J. Neill. *Antipatterns – Identification, Refactoring, and Management*. Auerbach Publications, 2006. ISBN 0849329949.

[PRESS06] Roger S. Pressman. *Ingeniería de Software. Un enfoque práctico*. Mc Graw Hill, sexta edición, 2006. ISBN 8448132149.

[AKROYD96] Michael Akroyd. *AntiPatterns: Vaccinations against Object Misuse*. Object World West, 1996.

[MARTIN90] James Martin. *Rapid Application Development*. MacMillan Publishing Co., New York, 1991. ISBN 0023767758.

[MCC95] Jim McCarthy. *Dynamics of Software Development*. Microsoft Press, 1995. ISBN 0735623198.

[AEDW06] Amr Elssamadisy, David West. *Adopting Agile Practice – An Incipient Pattern Language*, 2006. Paper.

[DEEPAK07] Deepak Khazanchi, Ilze Zigurs. *An Assessment Framework for Discovering and Using Patterns in Virtual Project Management*. Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[DEMARCO99] Tom de Marco, Timothy Lister. *Peopleware: productive projects and teams*. Dorset House Publishing Co, 1999. ISBN 0932633439.

[JACOBSON99] Ivar Jacobson, Grady Booch and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999. ISBN 0201571692.

[BUSCH96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern orinented Software Architecture - A System of Patterns*. Wiley and Sons, 1996. ISBN 0471958697.

[PP06] Deepak Khazanchi, Ilze Zigurs. *Patterns of Effective Management of Virtual Projects: An Exploratory Study*. Project Management Institute, 2005. ISBN 1930699832.

[PALMER97] Palmer, J.W. and C. Speier. *A Typology of Virtual Organizations: An Empirical Study*. Proceedings of the 1997 Americas Conference on Information Systems, Association for Information Systems, 1999.

[SHEN98] Shenhar, A. J. *From Theory to Practice: Toward a Typology of Project Management Styles*. IEEE Transactions on Engineering Management, 45(1), pp. 33-48, 1998.

[SCHW01] Ken Schwaber, Mike Beedle. *Agile Software Development with Scrum* Prentice Hall, 2001. ISBN 0130676349.

[ROYCE70] Winston Royce. *Managing the Development of Large Software Systems*. Procedings, IEEE WesCon, p.1-9, 1970.

[VOLTER02] Völter, M. *Hope, Belief and Wizardry – Three Different Perspectives on Project Management*. EuroPLoP 2002, Seventh European Conference on Pattern Languagesof Programs, Irsee, Germany, 2002.

[ZIGURS07] Zigurs, I., R. Evaristo, and B. Katzy. *Collaborative Technologies for Virtual Project Management*. Academy of Management, Washington, DC. 2001. Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[DUBE04] Dubé, L., and G. Paré, D. J. Pauleen. *The Multi-Faceted Nature of Virtual Teams*. Virtual Teams: Projects, Protocols, and Processes, Idea Group Publishing, Hershey, PA, pp. 1-39, 2004.

[KRUCH00] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 2 edition, 2000. ISBN 0201707101.

[PINS05] Pinsonneault, A., and O. Caya. *Virtual Teams: What We Know, What We Don't Know*. International Journal of e-Collaboration, 1(3), pp. 1-16, 2005.

[POWELL04] Powell, A., G. Piccoli, and B. Ives. *Virtual Teams: A Review of Current Literature and Directions for Future Research*. The Database for Advances in Information Systems, 35(1), pp. 6-36, 2004.

[NUNA91] Nunamaker, J. F., A. R. Dennis, J. S. Valacich, D. R.Vogel, and J. F. George. *Electronic Meeting Systems to Support Group Work*. Communications of the ACM, 34(7), pp. 40-61, 1991.

[ZIGURS98] Zigurs, I., and B. Auckland. *A Theory of Task/Technology Fit and Group Support Systems Effectiveness*. MIS Quarterly, 22(3), pp. 313-334, 1998.

[ANTTI07] Antti Nurmi, Pentti Marttiin, Matti Rossi. *Communication Patterns over the Project Life Cycle – Evidence from a Virtual Project Exercise*. Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[KHA05] Khazanchi, D., and I. Zigurs. *Patterns of Effective Management of Virtual Projects: An Exploratory Study*. Project Management Institute, Newtown Square, PA, 2005.

[CROW91] K. Crowston. *A Coordination Theory Approach to Organizational Process Design*. Organization Science, 8(2), pp. 157-175, 1991.

[GRANT96] Grant, R. M.. *Toward a Knowledge-Based Theory of the Firm*. Strategic Management Journal, 17, Winter Special Issue, pp. 109-122, 1996,

[HEND92] Henderson, J. C., and S. Lee. *Managing IS Design Teams: A Control Theories Perspective*. Management Science, 38(6), pp. 757-777, 1992.

[KIRSCH96] Kirsch, L. J. *The Management of Complex Tasks in Organizations: Controlling the Systems Development Process*. Organization Science, 7(1), pp. 1-21, 1996.

[KERZNER03] Kerzner H. *Project Management*. Wiley Octava edición, 2003. ISBN:

[BECK99] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000. ISBN: 0201616416.

[GILB88] Tom Gilb. *Principles of Software Engineering Management*. Addison-Wesley, 1988. ISBN 0201192462.

[AEDW06] Amr Elssamadisy, David West. *Adopting Agile Practice – An Incipient Pattern Language*, 2006. Paper.

[MOY89] Moynihan T et al. *Riskman 1: A prototype Tool for Risk Análisis for computer Software*. 3rd International Conference on Computer-Aided Software Engineering, 1989.

[LOCK08] Arden C. Lockwood. *The Project Manager´s Perspective on Project Management Software Packages*. Survey Report for PMI. 2008.

[SAD1] L. Domínguez, S. Jaureche. *Switch de Pagos – SAD*. Inco – CPAP, 2007

[SAD2] Perovich y Vignanga. *Reserva hotelera – SAD*. Inco – CPAP, 2005.

[KRU95] P.Kruchten. *Architectural Blueprints—The "4+1" View Model of Software Architecture*. Rational Software Corp. IEEE Software 12 (6), 1995.

[LARMAN99] Larman, Craig. *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 1999. ISBN 0130925691

[HOON08] Young Hoon Kwak, Frank T. Anbari. *Impact on Project Management of Allied Disciplines: Trends and Future of Project Management Practices and Research*. Project Management Institute, 2008. ISBN: 9781933890456

[CAS07] *Curso Arquitectura de Software*. Inco – CPAP, 2007.

[HSMITH00] Highsmith, J.A. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House, 2000. ISBN 0-932633-40-4.

[NEP08] Tam´as Nepusz:. *Data Mining in Complex Networks: Missing Link Prediction and Fuzzy Communities*. Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary, 2008.

[NEW07] Newman M.E.J. *Detecting community structure in networks. Department of Physics and Center for the Study of Complex Systems*. University of Michigan, 2007.

[CRAW04] Lyn Crawford. *Global body of Project Management knowledge and Standards*. Wiley paper, 2008.

[DEAN07] Dean LeffingwellMay, *Scaling Software Agility: Best Practices for Large Enterprises* Addison-Wesley Professional , 2007. ISBN 0321458192.

[DEAN09] Dean LeffingwellMay. *Patterns, Anti-patterns and the Next Wave of Enterprise Agile Adoption*. Paper, 2009.

[HERBER08] Hermann Löh, Bernhard Katzy. *Project Management and Communication Patterns in Innovation Projects*. Paper, 2008.

[VAL08] Antti Välimäki. *Patterns for Distributed Scrum –a Case Study*. Paper, 2008.

[GRA06] Graeme Thickins. *"Email Mining": Emerging New Business Application*. Intradyn. Paper, 2006.

[AMBLER08] Scot W. Ambler. *The Portfolio Management Discipline: Scaling Agile Software Development*. Paper, 2008.

[ASH04] Ashok Reddy. *Project Portfolio Management (PPM): Aligning business and IT*. IBM Paper, 2004.

[MVC] http://www.martinfowler.com/eaaDev/uiArchs.html [Ult.V: Mayo 2009]

[CBURN05] http://alistair.cockburn.us/crystal/articles/doi/declarationofinterdependence.htm [Ult.V: Mayo 2009]

[APAT] http://c2.com/cgi/wiki?AntiPatternsCatalog [Ult.V: Mayo 2009]

[PAT09] http://www.iaria.org/conferences2009/CfPPATTERNS09.html [Ult.V: Mayo 2009]

[AGILE01] http://agilemanifesto.org/ [Ult.V: Mayo 2009]

[AMBER] http://www.ambysoft.com/unifiedprocess/agileUP.html [Ult.V: Mayo 2009]

[CBURN08] http://alistair.cockburn.us/index.php/

From_Agile_Development_to_the_New_Software_Engineering [Ult.V: Mayo 2009]

[CRISP01] http://www.crisp-dm.org/ [Ult.V: Mayo 2009]

[CRISP02] http://www.dataprix.com/modelo_CRISP-DM [Ult.V: Mayo 2009]

[SAD] http://www.cs.toronto.edu/~wl/teach/407/2002/rup-sad.html [Ult.V: Mayo 2009]

[UML03] http://www.omg.org/uml. [Ult.V: Mayo 2009]

[COAL] http://www.fing.edu.uy/inco/grupos/coal/ [Ult.V: Mayo 2009]

[RUP03] http://www.rational.com/rup [Ult.V: Mayo 2009]

[MIME01] http://en.wikipedia.org/wiki/MIME [Ult.V: Mayo 2009]

[RFC5322] http://tools.ietf.org/html/rfc5322 [Ult.V: Mayo 2009]

[NEU01] http://igraph.sourceforge.net/index.html [Ult.V: Mayo 2009]

[KEY01] http://www.keyhubs.com/CaseStudies.aspx [Ult.V: Mayo 2009]

[PAJ01] https://pajek.imfm.si/doku.php?id=pajek [Ult.V: Mayo 2009]

[APM01] http://www.apm.org.uk/ [Ult.V: Mayo 2009]

[PRINCE2] http://www.prince2.com [Ult.V: Mayo 2009]

[OGC] http://www.ogc.gov.uk/ [Ult.V: Mayo 2009]

[PMAG] http://www.pmaj.or.jp [Ult.V: Mayo 2009]

[SMALLTALK] http://www.smalltalk.org/ [Ult.V: Mayo 2009]

[PMICM] http://www.pmi.org.uy [Ult.V: Mayo 2009]

[PMI] http://www.pmi.org [Ult.V: Mayo 2009]

[OPM3] http://opm3online.pmi.org/ [Ult.V: Mayo 2009]

[ROSS07] http://agilemanager.blogspot.com/2007/04/patterns-and-anti-patterns-in-project.html [Ult.V: Agosto 2009]

.