Universidad Carlos III de Madrid

TESIS DOCTORAL

PATRONES DE PROYECTOS PARA GESTIONAR EL CONOCIMIENTO EN ORGANIZACIONES DE DESARROLLO SOFTWARE

Autor:
Diego Martín de Andrés

Directores:
Dr. D. Javier García Guzmán
Dr. D. Juan Bautista Llorens Morillo

DEPARTAMENTO DE INFORMÁTICA
Universidad Carlos III de Madrid

Leganés 2012

TESIS DOCTORAL

PATRONES DE PROYECTOS PARA GESTIONAR EL CONOCIMIENTO EN ORGANIZACIONES DE DESARROLLO SOFTWARE

Autor: D. Diego Martín de Andrés

Directores: Dr. D. Javier García Guzmán
Dr. D. Juan Bautista Llorens Morillo

Firma del tribunal calificador:

Firma

Presidente:

Vocal:

Secretario:

Calificación

Leganés, a de 2012

A mi padre...

Los programas feos son como los puentes feos: suelen ser mucho más propensos a caerse que los bonitos, porque la forma en la que los humanos percibimos la belleza (y especialmente los ingenieros) está íntimamente relacionada con nuestra habilidad de procesar y entender la complejidad.

-- Eric S. Raymond

RESUMEN

La mejora de procesos de desarrollo software en las organizaciones de desarrollo software es una tarea muy compleja que puede ser resuelta usando estrategias de gestión del conocimiento. En esta área, la definición y el uso de buenas prácticas en la ingeniería del software es una propuesta válida para aplicar estrategias de gestión del conocimiento en organizaciones de desarrollo software. Una de los principales problemas para la aplicación efectiva de patrones de procesos en la industria de desarrollo software es la dificultad de la formalización del conocimiento sobre los procesos de desarrollo usando estas aproximaciones.

Esta tesis doctoral presenta un framework para la gestión de patrones de proyectos de desarrollo software llamado sdpFramework. Este framework es capaz de formalizar el conocimiento sobre proyectos de desarrollo software incluyendo experiencia previa de ingenieros de software, metodologías de desarrollo, marcos de referencia y lecciones aprendidas. Este framework está compuesto por:

- Un modelo llamado sdPP; compuesto por los elementos de conocimiento necesarios para dar cobertura a las necesidades de conocimiento de los ingenieros de software.
- Una metodología; que cubra las fases del ciclo de vida del conocimiento sobre patrones de proyecto.
- Una plataforma tecnológica llamada sdpReuser; capaz de gestionar los sdPPs durante las fases del ciclo de vida de los patrones de proyecto.

Un caso de estudio embebido en dos partes se ha llevado a cabo durante la investigación de esta tesis doctoral:

La primera parte fue un estudio empírico en la Universidad Carlos III de Madrid, donde doce ingenieros junior de desarrollo software usaron los patrones de proyectos descritos en este trabajo de investigación. Las evidencias y resultados obtenidos durante la ejecución del estudio empírico indican que la corrección en la formalización de los patrones de proyectos depende de la relevancia de las referencias bibliográficas usadas para su creación, la implementación de estrategias para compartir conocimiento entre el personal involucrado y la experiencia previa en las áreas de negocio relacionadas con los sistemas de información desarrollados.

En la segunda parte se realizó una validación experimental donde 48 ingenieros de software aplicaron sdpFramework. De los resultados obtenidos, se estudió los factores que ayudan a mejorar la calidad de los productos de desarrollo software usando la propuesta sdPP. Se analizó el esfuerzo necesario para implementar las actividades propuestas por sdPP para introducir el conocimiento de un sdPP en un proyecto de desarrollo. Para finalizar se evaluó la utilidad de los elementos de conocimiento de sdPP en cada una de las fases de su ciclo de vida.

ABSTRACT

The improvement of software processes within software development organizations is a very complex task that can be solved by following knowledge management strategies. In this area, the definition and use of best practices is a proven approach to apply knowledge management strategies in software development organizations. One of the main burdens for the effective application of process patterns in the software industry is the difficulty of formalizing knowledge about the development process themselves.

This doctoral thesis presents sdpFramework, a framework to manage software project patterns. This framework is capable of formalizing knowledge regarding software development projects, including software engineers' previous experience, development methodologies, reference frameworks and lessons learnt. The framework is composed of:

- A model called sdPP, made up by the knowledge elements necessary to cover the knowledge needs of software engineers.
- A methodology that covers the phases of the knowledge life-cycle about project patterns
- A technological platform called sdpReuser, capable of managing the sdPPs during the phases of project patterns' life cycle.

An embedded case study was carried out during the research of this thesis.

The first part was an empirical study at the University Carlos III of Madrid, where twelve junior software engineers used the project patterns described in this research work. The evidences and findings obtained during from this empirical study indicate that the correctness of the project pattern formalizations depends on the relevance of the bibliographic sources used for their creation, the implementation of strategies to share knowledge among the personnel involved, and the previous experience in the business areas related to the information systems being developed.

In the second part an experimental validation was carried out, where 48 software engineers applied the sdpFramework. From the results obtained, we studied the factors that help improving the software product quality when using the sdPP proposal. We analyzed the effort required to implement the activities proposed by sdPP in order to introduce the knowledge about an sdPP into a development project. Finally, we evaluated the usefulness of the sdPP knowledge elements in each of the phases of its life cycle.

AGRADECIMIENTOS

El primer y el mayor de los agradecimientos son para mi director de tesis, Javier García Guzmán; sin él esta tesis no habría salido adelante y no se habría terminado. Gracias por la paciencia derrochada en el desarrollo de esta tesis, gracias por ayudarme en todos los problemas que han surgido en el desarrollo de esta tesis y gracias por ayudarme con los problemas personales.

Quiero agradecer toda la ayuda ofrecida por Antonio Amescua, ya que él ha dirigido esta tesis de forma indirecta mereciéndose estar entre los directores de ella, sin su ayuda y sin todas sus críticas constructivas esta tesis tampoco habría sido finalizada.

Quiero agradecer a Paloma Martínez el incluirme en el proyecto "*GPS: Plataforma de gestión de procesos software: Modelado, reutilización y medición.*" el cual ha sido el origen de esta tesis doctoral.

Además quiero agradecer el enorme esfuerzo que Julián Urbano ha dedicado a educarme en los métodos, técnicas y herramientas estadísticas que he necesitado para poder realizar la experimentación de esta tesis doctoral y los artículos relacionados.

Por último quiero agradecer a los grupos de investigación SEL-UC3M y Knowledge Reuse, con los que he podido trabajar y desarrollar esta tesis doctoral.

Gracias de todo corazón.

DEDICATORIAS

Son muchas las personas a las que quiero dedicar esta tesis doctoral que me han ayudado facilitando el sinuoso desarrollo de la misma.

La primera persona a la que quiero dedicar esta tesis, como bien reza la dedicatoria especial, es a mi padre José Antonio Martín, ya que desde el día que nací me ha ayudado a desarrollarme como persona fomentando mi formación y facilitando que pueda hacerlo; además me ha procurado cariño y comprensión con todos los problemas personales por lo que he pasado en lo que llevo de vida, sin él esta tesis no existiría. También quiero dedicar esta tesis a mi madre María José de Andrés por haberme cuidado desde pequeño preocupándose por mi bienestar. Quiero dedicar esta tesis a mis abuelos Marcelino Martín † y Ascensión Garrido por cuidarme e inculcarme valores desde muy pequeño.

Además quiero dedicar esta tesis doctoral a mis primos Cristina, Elena Martín, David, Israel, Raúl, Roberto, Juan José, Elena de Andrés, Nacho y Adrián.

Dedico esta tesis a Noemí García, quien fue mi compañera sentimental durante catorce años y que soportó la mayor parte de los años de la realización de esta tesis. También quiero dedicar mi tesis doctoral a Laura López, que sin esperar nada a cambio me ayudó a salir a flote cuando más lo necesitaba.

Quiero dedicar esta tesis a todos los compañeros de trabajo de la Carlos III que se convirtieron en amigos; a Julián Urbano, única persona en aparecer en los agradecimientos y dedicatorias de esta tesis, por ser un compañero incansable, mejor amigo, insaciable jugador de Quake y devorador de durums dobles; a Eduardo Barra por ser tan buen amigo y enseñarme a ver las cosas a la "mexicana"; a Mónica Marrero por comprenderme en todos los problemas que he tenido en la universidad y disfrutar de los mismos placeres de la vida: cine y aceitunas; a Jorge Morato por ser el primero en acogerme en el grupo KR y por compartir las mismas formas de pensar y de vivir; a Sonia Sánchez por ser tan generosa, predispuesta a ayudarme e incluirme en el proyecto biblio-dominicana; a Maricruz Valiente por tener el coraje de cambiar de universidad cuando las cosas iban mal; a Alberto Heredia por resolverme las dudas en el depósito de la tesis. A Miguel Ángel Zurdo por ser capaz de reírse de todo y ver la vida de esa forma tan divertida; a Álvaro Parra por ser incluso más "freak" que yo y ser tan creativo. A mi amigo griego Yorgos Andreadakis por compartir esa forma tan divertida de ver la vida; a Julio Encinas y Luis María Alonso por ser buenos compañero e increíbles desarrolladores de software que siempre me

han echado un cable cuando lo he necesitado; a José Miguel Fuentes por ser el primero en instruirme en el diseño basado en componentes.

Además me gustaría dedicar esta tesis doctoral a todos mis amigos que me han acompañado durante toda mi vida y en especial en la realización de esta tesis doctoral, han sido muchos y cada uno de ellos se merece al menos una hoja. A José María Egido, el mejor amigo que uno puede tener, capaz de escucharme en las horas más bajas y capaz de ponerse un mono de obra y a hacer las rozas en mi casa. A su compañera María Jesús González por ser tan generosa y gentil, especialmente cuando Egido y yo nos metemos en un lío y a su hermano Mariano González, por ser tan amable y divertido.

A Francisco José Martín; por ser el amigo más templado y sereno que conozco, por acordarse siempre de mí; a su mujer Sara García por dar ese toque de pasión a todo lo que le rodea. También quiero dedicar mi tesis doctoral a mi amigo Jesús Gómez (Chuso) que siempre me ofrece un punto de vista distinto al que yo veo; a su mujer Silvia Gómez por ser una de mis mejores amigas y por esa fuente de cariño inagotable y a Miguel Gómez, hijo de ambos, por empeñarse en que yo soy su tío.

Dedico esta tesis a Diego de Miguel (Maldini) amigo fiel, ecuánime, incansable y divertido; a Enrique Ramos, el gran erudito del grupo y con un pensamiento 100% independiente; a Julio Fernando Borreguero, mi gran amigo desde la EGB y compañero de aventuras donde él es Tintín y yo el capitán Haddock.

A Arturo Piñuela, compañero de miles de batallas en la Universidad Complutense y en la Carlos III, por mostrarme el funcionamiento no técnico del mundo y por su especial cariño; a Emilio García por ser tan comprensivo, su infinita simpatía y ser un autentico confidente; a Javier Castro por implantar esa consciencia de grupo que todavía nos mantiene unidos; a Mariano Velasco por defender su opinión con fuerza y ser siempre tan considerado.

Quiero hacer una dedicatoria especial a mi gran amigo mexicano Hugo Mitre que conocí en las clases de doctorado y gimnasio de la UC3M; él me ha enseñado otra forma de ver la vida, el valor de la familia y de los amigos; a su mujer Elisabeth Araujo por ser tan buena compañera y hacernos esas comidas mexicanas excepcionales que no me cansaré de degustar en compañía de ambos.

También dedico esta tesis a Ireneusz Bator, que empezó a trabajar en la construcción de mi casa y se convirtió en un buen amigo capaz dar una ingeniosa solución a cualquier tipo de problema constructivo o emocional.

Antes de terminar quiero dedicar este trabajo a Marcus † y Stutinky esos dos gatetes que siempre me han estado observando desde la silla mientras aporreaba un teclado y blasfemaba contra la pantalla; ellos siempre han estado ahí, a mi lado, mirando…

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO DE TÉRMINOS

Diagrama de actividad En el Lenguaje de Modelado Unificado, un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un Diagrama de Actividades muestra el flujo de control general.

Framework En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GUI (Graphical User Interface) La interfaz gráfica de usuario es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz

HTML (HyperText Markup Language) Lenguaje de marcado de hipertexto, hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes.

JCR (Journal Citation Reports) Es una publicación anual que realiza el Institute of Scientific Information, miembro de la empresa Thomson Scientific. Proporciona información sobre revistas científicas del campo de las ciencias aplicadas y sociales.

Metadatos (Del griego μετα, meta, 'después de, más allá de' y latín datum, 'lo que se da', «dato»), literalmente «sobre datos», son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos, llamado recurso. El concepto de metadatos es análogo al uso de índices para localizar objetos en vez de datos. Para varios campos de la informática, como la recuperación de información o la web semántica, los metadatos en etiquetas son un enfoque importante para construir un puente sobre el intervalo semántico.

Not To-do Tarea que se debe asegurar que no se debe realizar, no es modelable en un workflow (Ver sección 3.2).

Not To-dos Conjunto de "Not to-do".

PAL (Process Assets Library) es un repositorio de documentos y herramientas que pueden ponerse a disposición de los equipos de proyectos para facilitar el desarrollo de un proyecto y el cumplimiento de los estándares de software. Este repositorio se actualiza constantemente, tanto en los nuevos documentos y herramientas, que da como resultado una retroalimentación directa para el usuario.

Productflow Flujo de productos. Descripción del flujo de los productos que existen entre las actividades (Ver sección 3.2).

RDF (Resource Description Framework) El Marco de Descripción de Recursos (del inglés Resource Description Framework, RDF) es un framework para metadatos en la World Wide Web (WWW), desarrollado por el World Wide Web Consortium (W3C). Es un lenguaje de objetivo general para representar la información en la web (un metadato data model). Es una descripción conceptual.

Refactoring La refactorización es una técnica de la ingeniería de software para restructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

RUP (Rational Unified Process) El Proceso Unificado de Rational es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM. Junto con el Lenguaje

	Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.
SCM	(Software Configuration Management) Gestión de configuración de software es una especialización de la Gestión de configuración a todas las actividades en el sector del desarrollo de software.
SCRUM	Scrum es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software. Aunque Scrum estaba enfocado a la gestión de procesos de desarrollo de software, puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas: Scrum de Scrums.
SDP	(Software development process) Proceso para el desarrollo de software, también denominado ciclo de vida del desarrollo de software es una estructura aplicada al desarrollo de un producto de software
sdpFramework	Marco de conocimiento ofrecido en esta tesis doctoral formado por un modelo de datos sdPP, una metodología y una herramienta para su gestión llamada sdpReuser (Ver sección 3.1.1).
sdPP	(Software development project pattern) Patrones de proyectos de desarrollo software, modelo de conocimiento que se ofrece en esta tesis doctoral que modela los principales elementos de conocimiento referente a los proyectos de desarrollo software (Ver sección 3.2).
sdpProcessModel	Conjunto de procesos que se ofrecen en esta tesis doctoral para gestionar el conocimiento albergado dentro de un SDPP (Ver sección 3.3)
sdpReuser	Herramienta desarrollada para esta tesis doctoral que forma parte del marco sdpFramework. (Ver sección 3.4).
SDSS	(Software Development Small Settings) Pequeñas

empresas de desarrollo software.

SPEM (Software & Systems Process Engineering Metamodel Specification) Metamodelo para la ingeniería de procesos de desarrollo software; es un lenguaje de modelado de procesos basado en UML y respaldado por el OMG (Object Management Group). Ha sido desarrollado específicamente para el modelado y la difusión de los procesos de desarrollo de la ingeniería de software. Actualmente se encuentra en la versión 2.0 (OMG, 2008)

To-do Tarea a realizar pero que no puede ser modelado en un workflow (Ver sección 3.2).

To-dos Conjunto de "to-do".

UML (Unified Modeling Language) Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. (OMG, 2007)

WBS (Work Breakdown Structure) Estructura desglosada de trabajo. En gestión de proyectos una descomposición jerárquica orientada al entregable, del trabajo a ser ejecutado por el equipo de proyecto, para cumplir con los objetivos de éste y crear los entregables requeridos, con cada nivel descendente del WBS representando una definición con un detalle incrementado del trabajo del proyecto. El WBS es una herramienta fundamental en la gestión de proyectos

Workflow O flujo de trabajo, es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace

seguimiento al cumplimiento de las tareas. Generalmente los problemas de flujo de trabajo se modelan con redes de Petri.

XP (eXtreme Programming) La programación extrema es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

INTRODUCCIÓN

Tabla de contenidos: Introducción

1: INTRODUCCIÓN

1.1 Contexto

Esta tesis doctoral está ubicada bajo el marco de la Ingeniería del Software, más concretamente en el área de la mejora del proceso de desarrollo software.

El desarrollo de software es un proceso donde se necesita la gestión del conocimiento de forma intensiva donde los principales factores que evitan la mejora de la productividad son la repetición, el exceso de información y un uso ineficiente del conocimiento por parte de los ingenieros de software. (Verma & Tiwari, 2009).

Con el fin de reducir la complejidad de la aplicación de los principios de la gestión del conocimiento a la ingeniería del software y a la mejora del proceso de desarrollo software, hay varios estudios sobre el uso de las prácticas eficientes en organizaciones de desarrollo software (Sanchez-Segura et al., 2010)

A través de la literatura analizada los autores a menudo describen un modelo de patrones (Coplien, 2004; Fowler, 1997; Gamma et al., 1995; Buschmann et al., 2007; Ambler, 1999; Appleton, 1997; Sommerville, 2004) etc. y un conjunto de patrones creados bajo ese mismo modelo. Pero en la literatura no se ha encontrado suficientes trabajos que hablen de la facilidad que ofrece el modelo para formalizar nuevos patrones y agrupar patrones aplicables a un mismo proyecto, los factores que afectan a la hora de realizar una buena formalización, la facilidad que ofrece el modelo para ser adaptado para ajustarse a las necesidades de un proyecto concreto ni un estudio sobre las mejoras obtenidas con el uso de patrones de procesos de desarrollo software.

Los beneficios de la mejora de procesos (Herbsleb et al., 1994) son:

- Disminuye el coste del producto;
- Aumenta la productividad;
- Reducción en los tiempos de entrega;
- Mejora en la calidad;
- Disminuye el tiempo de retorno de la inversión.

El ciclo de vida de un patrón es complejo, largo e intervienen muchas personas, desde los creadores del patrón hasta los usuarios o las personas que

lo adaptan. Para que el conocimiento pueda aplicarse de forma efectiva y eficiente es necesario que los patrones cumplan unos ciertos requisitos:

- Un modelo de conocimiento que se mantenga durante todas las actividades del ciclo de vida de los patrones.
- Efectivo; que sea capaz de retener el conocimiento;
- Eficiente; que además requiera de pocos recursos;
- Útil para los usuarios; dependiendo de la actividad la utilidad se percibe de una forma distinta. Por ejemplo, en la fase de adquisición la utilidad consiste en la capacidad de retener la información necesaria; sin embargo, en la fase de uso la utilidad se entiende como la capacidad que tiene el patrón para resolver problemas en el desarrollo software.

1.2 Definición del problema y motivación

La mejora de procesos de software es una tarea muy compleja en las organizaciones de desarrollo de software que puede ser resuelta usando estrategias de gestión del conocimiento. La definición y uso de procesos de desarrollo software en forma de patrones es una solución que permite la formalización de conocimiento para resolver problemas recurrentes en el desarrollo del software.

En la actualidad existe mucho conocimiento sobre desarrollo software en forma de metodologías, buenas prácticas, lecciones aprendidas, marcos de referencia, etc. Toda esta información no se encuentra estandarizada, se encuentra en distintos formatos y, dependiendo del documento, la mayoría de ellos no son digitales. Hay varias propuestas en la literatura para intentar formalizar este conocimiento bajo un mismo modelo, pero es difícil encontrar mecanismos que permitan la formalización de todo el conocimiento sobre las buenas prácticas en el desarrollo de proyectos software. También es difícil encontrar mecanismos que faciliten la búsqueda por todos los elementos de conocimiento, independientemente del tipo que sean. Además no existen metodologías ni pautas que ayuden a la formalización de nuevo conocimiento o conocimiento ya existente; tampoco existen herramientas que ayuden a la gestión de este conocimiento para que pueda ser reutilizado.

En la literatura analizada, los autores ofrecen distintos tipos de patrones; por ejemplo patrones de análisis (Fowler, 1997), de diseño (Gamma, 1995), arquitecturales (Buschmann, 2007), de procesos (Ambler, 1999), de mejora del proceso (Appleton, 1997), de colaboración (Sommerville, 2004), etc. Estos trabajos suelen incluir un modelo de conocimiento y un conjunto de patrones creados siguiendo dicho modelo. Este conjunto de patrones están pensados

para resolver problemas recurrentes cada uno en su campo, pero los modelos con los que fueron creados no están diseñados para facilitar la creación de más patrones siguiendo sus modelos. En ningún momento se ha pensado en la facilidad que ofrece el modelo para crear más patrones, probablemente porque cuando se diseño el modelo se pensó en los patrones ya existentes y nunca en la posibilidad de que crear más. Es por ello que los modelos se deberían diseñar para crear más conocimiento, poder adaptarlos a las problemáticas de contexto donde se necesita utilizar, así como para evolucionar los ya existentes.

En la literatura analizada, los autores ofrecen un conjunto de patrones que no sobrepasa el número de cincuenta en el mayor de los casos. Si partimos de un número relativamente pequeño de patrones es fácil realizar búsquedas y también es fácil su distribución. Sin embargo, si ofrecemos un modelo que favorezca la generación de nuevo conocimiento en forma de patrones será necesario ofrecer una herramienta informática que facilite la creación de nuevo conocimiento, que ayude a la búsqueda de conocimiento y que distribuya dicho conocimiento para que pueda ser accesible por la comunidad para su reutilización. En cuanto a las búsquedas, aparecen nuevos problemas debido a que en la mayoría de los casos los campos de información del modelo son textos en lenguaje natural por lo que es necesario poder hacer búsquedas que puedan resolver preguntas en lenguaje natural. Otro problema con respecto a las búsquedas consiste en que ciertos elementos de conocimiento en el modelo son información estructurada, por ejemplo, workflows o diagramas de clases; realizar búsquedas en información estructurada es complicado por la naturaleza del conocimiento y la indexación de dicha información.

Por estas razones en esta tesis doctoral se propone un modelo de conocimiento que sea capaz de formalizar el conocimiento sobre metodologías, marcos de referencia, mejores prácticas, lecciones aprendidas, etc. sobre proyectos de desarrollo software en forma de patrones; un conjunto de procesos para cubrir las necesidades de conocimiento en todas sus fases basándose en los ciclos de vida propuestos en la literatura pero ofreciendo nuevas etapas y centrándose en la gestión de patrones y una herramienta que gestione este conocimiento en cada una de las fases que se proponen en el ciclo de vida de los procesos propuestos.

En resumen, esta tesis doctoral aborda los siguientes desafíos:

- Definir un modelo de conocimiento que admita procesar el conocimiento más importante de las buenas prácticas, lecciones aprendidas, marcos de referencias y metodologías de desarrollo de software. Además este modelo debe permitir crear nuevos patrones con el conocimiento tácito de la empresa; ya que obviamente se trata de conocimiento muy valioso.

- Definir un conjunto de procesos para la gestión de conocimiento relacionado con los patrones de proyecto, donde será necesario definir un ciclo de vida de este conocimiento para poder promover y distribuir el conocimiento sobre el desarrollo de software.
- Desarrollar una plataforma que soporte la creación de patrones de proyecto y que de un soporte integral en el ciclo de vida de los patrones de proyecto, como la creación, búsqueda, distribución, uso del conocimiento, etc.

1.3 Hipótesis de trabajo

La hipótesis que se va a tratar de validar en esta tesis doctoral es:

Si se ofrece un framework integral compuesto por un modelo de conocimiento para patrones de proyecto, un conjunto de procesos (aplicable desde la creación hasta el uso de los patrones de proyecto) y una herramienta para su gestión; entonces en las organizaciones de desarrollo de software:

- *Se facilitará la formalización de buenas prácticas de ingeniería de software aplicables a un proyecto de desarrollo, identificando los factores críticos para la elicitación del conocimiento.*
- *Se mejorará la calidad del producto software sin afectar significativamente en el tiempo de desarrollo.*
- *Se facilitará la identificación y aplicación de los elementos de conocimiento que ayudan a la adopción de las prácticas efectivas en las fases de adquisición, adaptación y utilización.*

1.4 Objetivos de investigación

El objetivo general de esta tesis doctoral consiste en mejorar el proceso de desarrollo software a través del uso de patrones de proyecto que encapsulan las mejores prácticas, lecciones aprendidas procedentes de marcos de referencia y metodologías de desarrollo. De esta manera cualquier organización dedicada al desarrollo de procesos software podrá arrancar procesos de mejora que de otra manera solo podrían hacerlo si tuvieran un grupo dedicado únicamente a los procesos o subcontratando dicha tarea. Para ello esta tesis ofrece un framework formado por un modelo de conocimiento para la gestión de patrones, que facilite su creación, modelado, adaptación y uso; y una herramienta para la gestión de dichos patrones en todas las fases del conocimiento.

El objetivo más específico que se pretende satisfacer con la realización de esta tesis doctoral es: Definir, soportar y validar un framework (modelo, conjunto de procesos y herramienta) para la mejora del proceso del desarrollo software usando patrones de proyecto software.

De este objetivo se definen varios objetivos específicos:

Objetivo 1: Definir un modelo de patrón de proyecto; que tenga cobertura para los elementos de información más utilizados en el dominio de la gestión de proyectos de desarrollo software, marcos de referencia, mejores prácticas y lecciones aprendidas. El modelo de conocimiento para los patrones de proyecto debe ser usable y facilitar la creación de nuevos patrones; por usable entendemos que sea efectivo, eficiente y que satisfaga las necesidades de gestión de conocimiento del usuario.

Objetivo 2: Definir un conjunto de procesos que describan el ciclo de vida de los patrones de proyectos; que cubra todas las fases de la gestión del conocimiento desde su creación hasta su uso. Para ello se analizará la literatura sobre los dominios de la mejora de procesos de desarrollo software y de la gestión del conocimiento; para crear un ciclo de vida de los patrones de proyecto y proponer nuevas fases no descritas en la literatura.

Objetivo 3: Desarrollo de una herramienta software que ayude a la gestión de patrones de proyecto. Se desarrollará una herramienta software con las siguientes características:

- Creación de patrones de proyecto
- Búsqueda de patrones en un repositorio de patrones de proyecto, por texto, patrones similares, por archivos, búsqueda gráfica, etc.
- Adaptación de patrones a las restricciones y necesidades de un proyecto de desarrollo software.
- Guiado en el uso de un patrón de proyecto durante el desarrollo del proyecto software.
- Distribución de los patrones a través de internet.

Objetivo 4: Validar la solución. Se quiere demostrar que utilizando sdpFramework propuesto se mejoran tres fases en el ciclo de vida de los patrones de proyecto:

- En la fase de elicitación de la información se consigue una mejora en la creación de patrones de proyectos favoreciendo el modelado de la información de una forma efectiva, eficiente y teniendo en cuenta la facilidad del usuario

- En la fase de adaptación de la información, los patrones de proyecto se pueden adaptar fácilmente a las necesidades y restricciones de un proyecto de desarrollo concreto.
- En la fase de uso de la información, (es decir de aplicar un patrón a un proyecto concreto) se mejora la calidad del producto software final sin que el uso de patrón de proyecto aumente significativamente el tiempo de desarrollo del proyecto software.

1.5 Aproximación a la solución

En esta tesis doctoral se ofrece una solución práctica para la gestión del conocimiento sobre procesos de desarrollo de software en forma de patrones de proceso de desarrollo software. Esta sección pretende introducir cómo se van a abordar los objetivos presentados en la sección anterior.

Objetivo 1, definir un modelo de patrón de proyecto.

Para acometer este objetivo de investigación se ha realizado un análisis del estado arte para recopilar los elementos de conocimiento más utilizados en las mejores prácticas, lecciones aprendidas, marcos de referencia y metodologías de desarrollo software; así como el conocimiento tácito sobre procesos de una organización de desarrollo software; se han estudiado los elementos de información de cada una de las propuestas en la literatura para comprender y analizar cuales son las que cumplen las condiciones de ser efectivas, eficientes; y que además satisfagan las necesidades del usuario.

Con esta información recopilada se ha diseñado un modelo de conocimiento de patrones de proyectos llamado sdPP (Software Development Project Pattern) que permita la creación de nuevo conocimiento formalizando nuevos patrones; ya que uno de los problemas más importantes encontrados en la gestión del conocimiento ha sido el referente a la facilidad de generar conocimiento nuevo (es decir, nuevos patrones) de los modelos de patrones de conocimiento en general y los patrones de proceso en particular. Por normal general, los trabajos en ingeniería de software sobre patrones (Carey & Carlson, 2002; Berczuk & Appleton, 2003; Sommerville, 2004; Landaeta et al., 2008; Isla-Montes & Gutiérrez-Vela, 2004; Surhone et al., 2010; Johnson et al., 2010) ofrecen un modelo de conocimiento, formado en su mayoría por un conjunto de atributos con información en lenguaje natural; y un conjunto de patrones diseñados bajo este modelo de conocimiento. En ningún trabajo analizado se estudia la facilidad para creación de nuevos patrones y conocimiento, la utilidad, efectividad y satisfacción del usuario del modelo analizado; así como la capacidad de poder agruparse varios patrones de proceso para crear uno nuevo patrón. *Es por ello, que en esta tesis doctoral se ofrece un modelo para la gestión de conocimiento referente a los patrones de proyecto de desarrollo software, que sea*

efectivo, eficiente, útil y que satisfaga las necesidades de información del usuario en todas las fases del ciclo de vida de los patrones de proyecto.

Objetivo 2, Definir un conjunto de procesos para el ciclo de vida del conocimiento.

Con este objetivo se pretende ofrecer un framework integral que ofrezca cobertura completa en el ciclo de vida del conocimiento. Para resolver este objetivo se ha realizado un estudio del estado del arte para analizar los modelos de ciclo de vida del conocimiento, estudiar sus fases, beneficios y defectos.

Con este marco metodológico, las organizaciones de desarrollo software pueden aplicar todo conocimiento existente sobre el desarrollo de software sin necesidad de ser especialistas ni tener un departamento especializado en procesos de desarrollo software; de esta manera, por ejemplo, las SDSS (Software Development Small Settings) son capaces de arrancar procesos de mejora continua.

Esta solución permitirá mejorar la gestión de conocimiento en todas sus fases; mejorando la creación, distribución, reutilización, adaptación y uso de conocimiento. Las actividades de gestión del conocimiento ofrecidas están basadas en las actividades de los principales modelos clásicas propuestas por (Bots & de Bruijn, 2002; Massey et al., 2002; Lindsey, 2002; Jennex & Olfman, 2004; Maier, 2007; Rowley, 2007) pero adaptadas para ajustarse a los requisitos y restricciones del dominio de los procesos de desarrollo software como vemos en la Figura 1

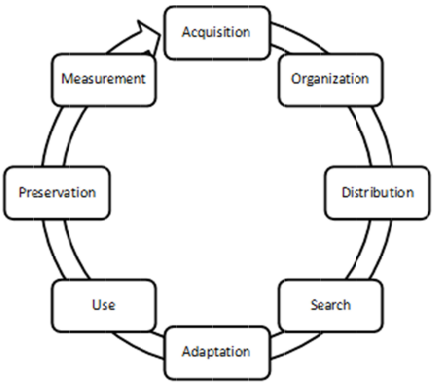


Figura 1 Ciclo de vida de los patrones de proceso

En la literatura existen varias prácticas y estrategias para la gestión del conocimiento (Véase capítulo 2) pero ninguna de ellas se ajusta exactamente a las necesidades de los procesos de desarrollo software en forma de patrones de proyectos de desarrollo software. Solamente se encuentran patrones de conocimiento orientados a los procesos o a productos, pero no definen en detalle ni una metodología, ni los elementos necesarios para ejecutar un

proyecto de desarrollo de software completo. *Por lo tanto, el framework definido en esta tesis doctoral propone un modelo de ciclo de vida de los patrones de proyecto que sea capaz de modelar la información más relevante de buenas prácticas, lecciones aprendidas, marcos de referencia más utilizados y metodologías de desarrollo; así como la capacidad de modelar las procesos tácitos de una empresa de desarrollo de software.*

Objetivo 3, Desarrollo de una herramienta software para soportar la gestión de sdPPs.

Una vez más se ha realizado un estudio del estado del arte, esta vez para analizar las herramientas que dan soporte a mecanismos de gestión del conocimiento; analizando los beneficios y carencias de las propuestas más utilizadas y difundidas. Debido a que en la literatura analizada no se ha encontrado ninguna herramienta que de soporte los procesos y el modelo de datos propuestos en esta tesis doctoral, se ha optado por desarrollar un framework software que de cobertura en todas las fases propuestas en el framework propuesto en esta tesis doctoral.

Otro de los problemas de los modelos de patrones actuales es que muy pocos tienen una herramienta informática que los apoye siendo muy complicado el funcionamiento correcto de ciertas actividades del ciclo de vida de los patrones, como por ejemplo en la distribución de información o en la búsqueda; siendo estas dos acciones muy importantes para la reutilización del conocimiento. Además la mayoría de los campos de información de los patrones están basados en lenguaje natural por lo que el proceso de búsqueda es mucho más complejo. *En esta tesis doctoral se va a ofrecer una herramienta para la gestión de patrones de proyecto de desarrollo software, que cubra todas las fases del ciclo de vida de los patrones de proyecto. Esta herramienta usará técnicas de procesamiento de lenguaje natural para poder indizar información sobre los patrones y poder hacer búsquedas en lenguaje natural, así como búsquedas gráficas por estructuras de ciertos elementos como los workflows, o búsquedas por similitud entre patrones.*

Objetivo 4, Validar la solución propuesta en esta tesis doctoral.

Para comprobar que la solución propuesta en esta tesis doctoral es válida, se han diseñado y se han ejecutado varios experimentos empíricos.

En el método empírico es necesario validar la hipótesis propuesta. En este apartado se va a describir los experimentos y casos de estudio diseñados para poder dar respuesta a cada objetivo de esta tesis.

La propuesta de validación de esta tesis consta de dos fases, cada una de ellas se van a evaluar una serie de objetivos.

Fase I: Validación experimental de la facilidad de modelado de patrones de proyecto y análisis de los factores que afectan en su uso. El objetivo de esta fase consiste en comprobar la facilidad de modelado de patrones de proyecto y

analizar los factores que afectan para saber cómo y cuándo aplicar un patrón de proyecto. Como se explicó en el apartado 3, este modelo de conocimiento ha sido diseñado para cubrir las necesidades de información cuando se ejecuta un proyecto software, pero también hay que tener en cuenta la efectividad, la eficiencia y la satisfacción del usuario cuando el patrón es modelado.

Para validar la facilidad de modelado de conocimiento se han diseñado los siguientes objetivos concretos:

- Paso 1: Comprobar la eficacia y eficiencia de los elementos de información del modelo de conocimiento para patrones de proyecto a la hora de modelar la información. Para esto se analizarán un conjunto de indicadores y mediciones relacionados con la eficacia y eficiencia. Estos indicadores están relacionados con la calidad en la formalización de diferentes modelos, así como la calidad de la bibliografía seleccionada y número de veces que se ha necesitado ayuda para resolver el problema.
- Paso 2: Evaluar la satisfacción del usuario que modelo patrones de proyecto, y para cada uno de los elementos de información de los patrones de proyecto. Para evaluar este objetivo se analizará la satisfacción de las personas que modelaron los patrones, primero de forma global y segundo por elemento de información.
- Paso 3: Analizar los factores que afectan a la hora de saber cuándo y cómo aplicar los patrones. Para ello se va a analizar distintos factores como la calidad de las fuentes bibliográficas, la utilidad de distintos elementos de información del patrón, etc.

Fase II: Validación experimental la facilidad de adaptación de los patrones y el aumento de calidad en el producto de desarrollo. El objetivo de esta fase consiste en comprobar la facilidad de adaptación de patrones de proyecto a los requisitos y necesidades de un proyecto de desarrollo concreto. En el capítulo 3 se explica que es necesario un modelo de ciclo de vida para la gestión de patrones de proyecto que ayude a la gestión del conocimiento de forma global. Se va a realizar una experimentación para analizar la facilidad de adaptación y uso de los patrones en un proyecto de desarrollo software.

- Paso 1: Analizar cuál es el grado de adaptación de cada uno de los elementos de información del modelo de los patrones de proyecto para adaptarse a las necesidades reales de un proyecto software. Para ello se va a realizar una experimentación donde usuarios del patrones van a tener que adaptar los patrones de proyecto a proyectos concretos de desarrollo software. Se para cada elemento de información del modelo se analizará la utilidad, la facilidad de

adaptación, lo intuitivo de la adaptación, el tiempo empleado en la adaptación, etc.

- Paso 2: Estudiar la utilidad de los patrones de proyecto cuando son aplicados a la ejecución desarrollo software. En esta fase se analizará la calidad de los productos generados después de un desarrollo, como es el proyecto software, así como la documentación técnica. Además se analizará el tiempo para aprender el patrón de proyecto, el tiempo empleado en su uso, el tiempo de desarrollo, etc.

1.6 Aportaciones de la investigación

Esta tesis doctoral ofrece una solución para la gestión de conocimiento sobre la gestión y desarrollo de proyectos de desarrollo software; las aportaciones que obtiene cualquier organización son: mejora en la creación de nuevo conocimiento en forma de patrones; factores que ayudan a saber como y cuando aplicar el conocimiento de patrones; aumento de la calidad del producto final; no aumento significativo del tiempo de desarrollo.

- Creación de nuevo conocimiento en forma de patrones; con el framework propuesto en la tesis se va favorecer la creación de conocimiento basado en patrones de proyecto ya que se ofrece un modelo usable, que cumple con las necesidades de información para poder modelar las buenas prácticas, lecciones aprendidas, marcos de referencia y metodologías de desarrollo. Permitiendo también el modela de la información tácita de una organización concreta.
- Análisis empírico que estudie los factores que ayudan a saber como y cuando aplicar el conocimiento de patrones; en la ejecución de esta tesis doctoral se ha realizado una experimentación empírica para poder analizar los factores que influyen en "cuándo" y el "cómo" aplicar los patrones de proceso. Con esta información se puede saber qué patrones aplicar y a qué proyecto.
- Análisis empírico que estudie el aumento de la calidad del producto final; en esta tesis doctoral se ha demostrado que utilizando los patrones de proyecto se consigue un aumento significativo en la calidad del producto desarrollado; así como en la documentación técnica desarrollada en cada proyecto.
- Análisis empírico que estudie que no aumenta significativamente el tiempo de desarrollo; con el uso de patrones de proceso no

produce un crecimiento significativo del tiempo de desarrollo. Por lo que se puede determinar que el uso de patrones de proyecto no incurre en un gasto significativo de recursos.

1.7 Método de investigación

1.7.1 Método y solución propuesta

El método de investigación elegido en esta tesis doctoral es el método empírico, siendo este uno de los más usados en el campo de la Ingeniería del Software. En esta tesis se ofrece una propuesta formal que se evalúa a través de estudios empíricos, tales como: Casos de estudio, estudios empíricos, experimentos y entrevistas.

La mejora del proceso de desarrollo software a través de patrones es un campo poco tratado en la literatura, si bien se habla de patrones de proceso (Carey, 2002; Ambler, 1998; Appleton, 1997), pero en ningún caso de patrones aplicables a un proyecto de desarrollo completo. De esta manera, este trabajo de investigación propone un nuevo marco de referencia para la gestión de patrones de proyecto, definiendo un ciclo de vida para los patrones de proyecto, así como, un nuevo modelo de conocimiento para poder gestionar esta información de forma efectiva y eficiente a través de cada uno de las fases del ciclo de vida definido.

Este marco de referencia y modelo de conocimiento, está soportado tecnológicamente por una herramienta software que da soporte a los patrones de proyecto en cada una de las fases del ciclo de vida de los patrones.

1.7.2 Planificación

Para alcanzar los objetivos planteados en esta tesis doctoral se han definido un conjunto de fases y actividades, tal y como se muestra en la Figura 2; donde se ha definido una estructura desglosada de trabajo (WBS, Work Breakdown Structure) para dar soporte a los objetivos específicos de la tesis mencionados en la sección 1.4

En la Figura 2 podemos ver gráficamente el WBS de la planificación de la presente tesis doctoral.

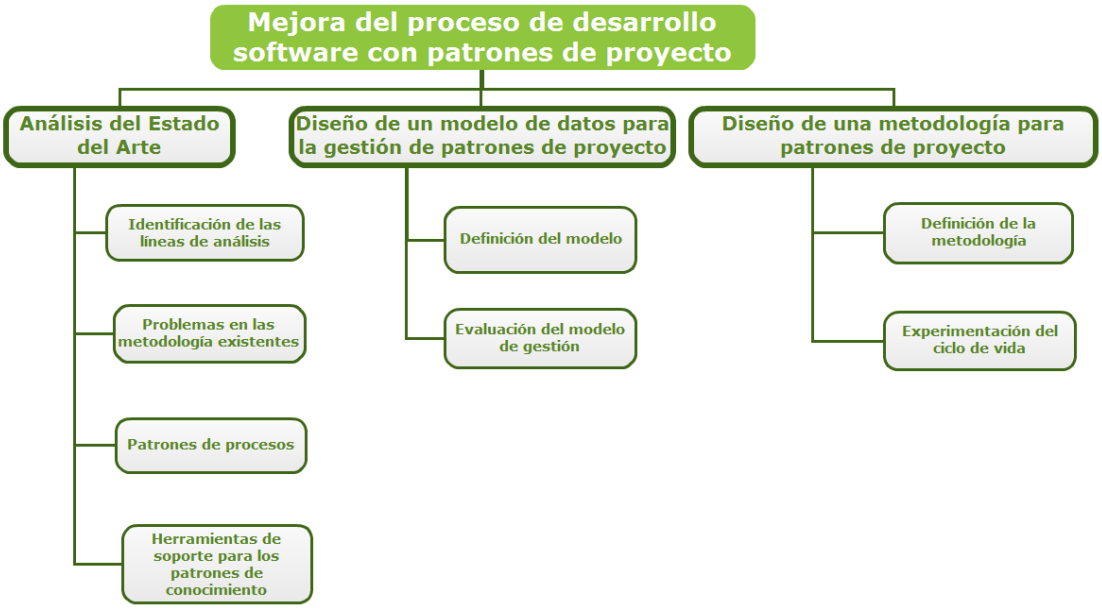


Figura 2 Planificación

1. Análisis del estado del arte: esta fase incluirá el estudio de la problemática presentada
    1. Identificación de las líneas de análisis: y líneas de investigación relacionadas con la gestión de patrones y más concretamente patrones de procesos, para poder profundizar en cada una de ellas.
    2. Problemas en las metodologías existentes: Análisis de las principales metodologías y procesos utilizados para la gestión de procesos y patrones de procesos.
    3. Patrones de procesos: Análisis y estudio de los principales patrones de procesos, con el objetivo de analizar sus puntos fuertes y sus deficiencias.
    4. Herramientas de soporte para los patrones de conocimiento, especialmente orientado a patrones de proyectos de desarrollo software. El objetivo de esta fase es buscar los puntos fuertes y deficiencias de las herramientas de gestión de patrones de conocimiento.
2. Diseño de un modelo de conocimiento para la gestión de patrones de proyecto. Esta fase incluirá la definición del modelo de conocimiento para la gestión de patrones de proyecto y la validación del mismo.

1. Definición del modelo. Se define un modelo de conocimiento para la gestión de patrones de proyecto que cumpla con las necesidades de esta tesis y que ayude en todas las fases del ciclo de vida de la gestión del conocimiento.
2. Evaluación del modelo de gestión. Se va a realizar una experimentación que sobre el modelo de conocimiento que demuestre la eficacia, la eficiencia y la satisfacción del usuario; del modelo propuesto.

3. Diseño de un conjunto de procesos para los patrones de proyecto. En esta fase se incluirá una propuesta de ciclo de vida para la gestión del conocimiento referente a patrones de proyecto
    1. Definición del ciclo de vida; en esta fase se van a realizar una definición de las fases del ciclo de vida necesarias para la gestión del conocimiento basado en patrones de proyecto.
    2. Experimentación de los procesos del ciclo de vida de los patrones de proyecto, se ha realizado una experimentación para validar los procesos y el modelo de conocimiento escogido en la fase 2

1.8 Estructura de la tesis doctoral

Esta tesis doctoral se estructura de la siguiente manera:

Capítulo 1. Introducción: Se trata del capítulo actual. En este capítulo se muestra el contexto de la investigación. Se marca el problema a partir de dicho contexto y se muestra la motivación de la investigación. Asimismo, se describen los objetivos de investigación y se presenta de forma somera una aproximación a la solución. Más adelante se detalla las aportaciones a la investigación y para concluir se describe la validación propuesta en esta tesis.

Capítulo 2. Estado del arte: Se presentará el estado del arte realizando una revisión crítica de los trabajos existente, comparando además dichas soluciones con la solución propuesta en esta tesis doctoral.

Capítulo 3. Solución propuesta: En este capitulo se define la solución propuesta en esta tesis doctoral para resolver los problemas expuestos en el capítulo 1.

Capítulo 4. Validación: Este capítulo describe la planificación, ejecución y el análisis de los datos obtenidos en la validación experimental realizada en este trabajo.

Capítulo 5. Conclusiones y trabajos futuros: Este capítulo mostrará las conclusiones de esta tesis doctoral, además se propondrán futuras líneas de investigación que han surgido tras la investigación realizada en esta tesis.

Capítulo 6. Bibliografía: Referencias literarias a los trabajos consultados y utilizados para el desarrollo de esta tesis doctoral.

ESTADO DEL ARTE

Tabla de contenidos: Estado del arte

2: ESTADO DEL ARTE

2.1 Introducción

En este trabajo de investigación se ha realizado un estudio de la literatura actual para tratar de analizar el estado actual de las principales áreas de conocimiento donde se apoya esta tesis doctoral.

Esta tesis doctoral se ha enfocado desde tres puntos distintos, el primero trata de investigar cuales son los ciclos de vida del conocimiento más utilizados y difundidos. El segundo campo de investigación pretende indagar entre los objetos de conocimiento más utilizados en ingeniería de software para la gestión de proyectos de desarrollo software. Mientras que el tercer campo de investigación analizará las tecnologías existentes sobre la gestión de conocimiento que pueden ayudar a la mejora de procesos de desarrollo software.

Los campos de investigación indicados en el párrafo anterior están relacionados directamente con áreas de investigación concretas. La primera área de investigación son los Patrones como objeto para la gestión de conocimiento en donde se ha realizado la mayor parte de investigación de esta tesis doctoral, ya que lo que pretende esta tesis doctoral es proponer una nueva técnica para poder desarrollar software de una forma efectiva y eficiente. Para ello se han analizado metodologías de desarrollo, marcos de referencia, buenas prácticas, lecciones aprendidas y métodos de gestión de conocimiento tácito en empresas.

Otra área de investigación es la Gestión del conocimiento: en esta área se ha estudiado la gestión del conocimiento a través de patrones, descubriendo deficiencias en la aplicación efectiva de dicho conocimiento. Para ellos se va a proponer un ciclo de vida integral para los patrones compuesto por las siguientes fases: Adquisición (el conocimiento es codificado y formalizado desde la información usando formularios o plantillas); organización (del conocimiento recopilado de acuerdo a cierta estructura establecida, relacionando elementos de conocimiento con otros); distribución (del conocimiento publicándolo de tal manera que sea accesible para cualquiera, normalmente a través de internet o intranet), búsqueda (del conocimiento relevante para ser usado; a través de palabras clave, metadatos o elementos gráficos), adaptación (del conocimiento para ajustarse al contexto donde se va a aplicar), uso (del conocimiento, siendo este el objetivo final de la gestión de

conocimiento a través de patrones), preservación (permitiendo a los usuarios del conocimiento actualizar el cuerpo del conocimiento, ya sea añadiendo nuevos patrones basándose en su propia experiencia o eliminando conocimiento irrelevante o desactualizado) y medición (analizando los patrones utilizados, buscados y modificados para llevar un control de posibles problemas y poder desplegar procesos de mejora continua).

Y por último, se han realizado investigaciones en el área de conocimiento, Mejora de procesos, donde parte de la investigación de esta tesis doctoral se ha realizado en esta área. Uno de los objetivos de esta tesis doctoral consiste en mejorar los procesos de desarrollo software ofreciendo una estrategia para la mejora continua de los procesos de una organización de desarrollo software.

Esta tesis doctoral presenta una nueva estrategia para la gestión de conocimiento sobre la gestión del conocimiento de las prácticas del desarrollo software permitiendo a las organizaciones de software permitiendo la mejora continua de su proceso.

2.2 Ciclos de vida en la gestión de conocimiento

Como afirmó (Freeze & Kulkarni, 2011) un objeto de conocimiento es un recurso complejo y no estático. Las organizaciones que quieran desarrollar y usar el conocimiento de forma eficiente deben tratar sus objetos de conocimiento según indiquen las fases de su ciclo de vida.

Existen varias investigaciones en el área de gestión de conocimiento donde se proponen varias fases. En 1995, Nonaka y Takeuchi (Nonaka & Takeuchi, 1995) ofrecieron por primera vez el término "ciclo de vida del conocimiento" y propusieron el modelo SECI (Socialización, Externalización, Combinación e Internalización). Birkinshaw (Birkinshaw & Sheehan, 2003) ofrece un modelo de tres fases mientras que Zack (Zack, 2002) describe un ciclo de vida de cinco fases.

Para la comparación de distintas propuestas en el estado del arte de la presente tesis doctoral se va a realizar un análisis de los modelos propuestos por Rus (Rus & Lindvall, 2002) y Nonaka (Nonaka & Takeuchi, 1995). Aunque existen más modelos en la literatura estos dos son los que más se ajustan a las necesidades de conocimiento necesarias para el propósito de esta tesis doctoral.

2.2.1 Modelo SECI

En Japón se propuso un nuevo modelo para el proceso de creación de conocimiento, dicho modelo fue desarrollada por Ikujiro Nonaka e Hirotaka

Takeuchi (Nonaka & Takeuchi, 1995). El objetivo del modelo SECI es entender la creación del conocimiento de forma dinámica y eficiente. Este modelo consiste en tres elementos:

- Ciclo de vida
- Contexto de la organización
- Activos de conocimiento

Cada uno de estos elementos interactúa con los otros de forma orgánica y dinámica. Los activos del conocimiento de la organización se activan y comparten el elemento "Ba" (Contexto de la organización) al tiempo que el conocimiento tácito propuesto por las personas es cambiado y aumentado por la dinámica del conocimiento a través de varios procesos:

- Socialización
- Externalización
- Combinación
- Internalización

La integración de los tres elementos debe hacerse siguiendo unas directrices marcadas, de modo que el resto de los componentes de la organización puedan aportan conocimiento de manera continua y estable.

A) Ciclo de vida

El conocimiento se crea a través de un proceso permanente de relaciones dinámicas, por una parte el conocimiento tácito y por otra el explicito. Las cuatro formas que convierten el conocimiento actúan en una espiral de creación de conocimiento. La espiral tendrá unas dimensiones cuando aumentan los niveles de organización, provocando nuevas espirales de creación de conocimiento.

Figura 3 El modelo SECI (Nonaka & Konno, 1999)

Socialización: Compartir el conocimiento tácito con una comunicación directa o de la experiencia compartida. Un ejemplo puede ser el aprendizaje.

Externalización: Desarrolla conceptos, que entran en el conocimiento tácito de combinación y que posibilitan su comunicación.

Combinación: De todos los elementos de conocimiento explícito: un ejemplo puede ser la construcción de un prototipo.

Internalización: Muy relacionada con el aprendizaje en acción, el conocimiento explícito se transforma en parte del inicio del conocimiento personal, por ejemplo los modelos mentales, y se transforma en un activo para el desarrollo de la organización.

B) Contexto organizativo

Este concepto (que podemos traducir como contexto, entorno o ecosistema) se puede definir como un contexto definido y compartido en el cual se interrelaciona el conocimiento, creado y utilizado con la interacción.

Ejemplos de las categorías, usando como ejemplo Seven-Eleven:

- Originando el Ba: el piso de la tienda, permite a la gente interactuar con cada uno de los otros y con los clientes.
- Dialogado Ba: el conocimiento tácito de los empleados del local se utiliza para crear pronósticos de ventas, en diálogo con cada uno de los otros.
- Sistematización del Ba: los pronósticos de ventas se prueban contra los resultados de ventas y se retroalimentan hacia los almacenes locales.
- Ejercitar el Ba: usando esta información, y comparándola con la realidad, el personal mejora sus habilidades y su capacidad de hacer pronósticos.

C) Activos de conocimiento

Son recursos específicos para la organización que son necesarios para crear valor añadido para la empresa. Son las entradas, salidas, y los índices de ponderación, de creación del proceso de conocimiento. Se puede ver un esquema de los tipos de activos de conocimiento en la Figura 4.

Para manejar con eficacia la creación y explotación del conocimiento, la organización tiene que "dibujar" su conjunto de activos de conocimiento. La catalogación por supuesto no es suficiente: los activos del conocimiento tienen dinamismo; los activos del conocimiento nuevo se pueden tener a partir de activos de conocimiento anteriormente existentes.

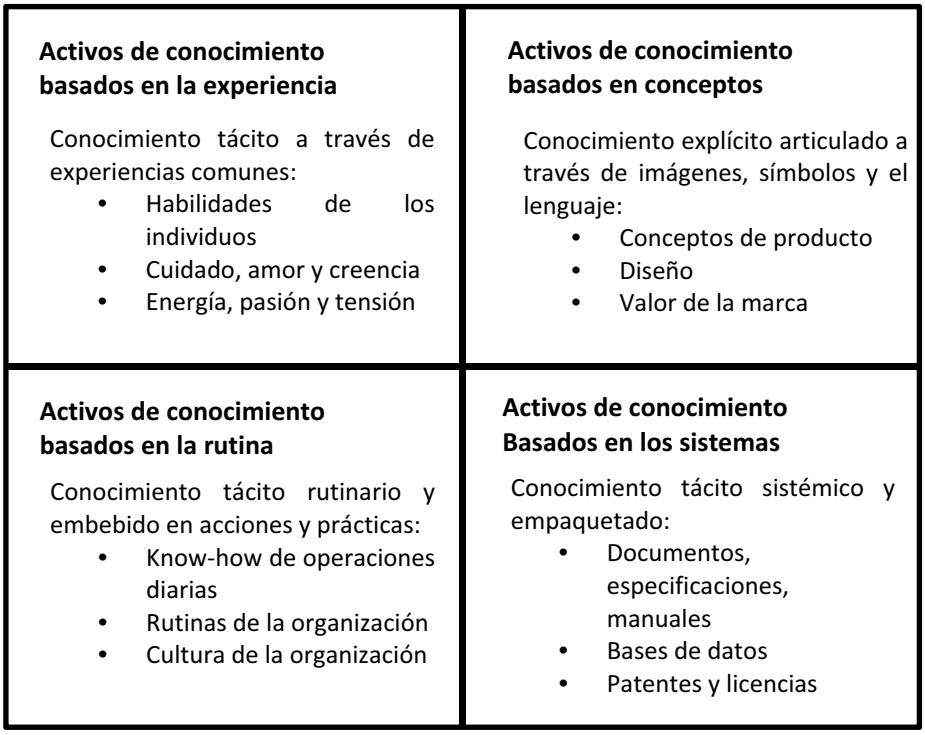


Figura 4 Las cuatro categorías de los activos de conocimiento

Fortalezas del modelo SECI:

- Constata la naturaleza dinámica del conocimiento y de la creación del conocimiento.
- Constituye un marco para la generación de procesos relevantes.

Limitaciones del modelo SECI:

- Basado en un estudio de las organizaciones japonesas, que confían en el conocimiento tácito: por lo general los empleados están con una compañía durante toda la vida.
- La horizontalidad del concepto: no permite dar saltos ni vaivenes entre el desarrollo de las actividades y no permite que la espiral gire hacia la izquierda.

2.2.2 Ciclo de vida en la gestión de conocimiento

Como estableció Rus (Rus & Lindvall, 2002), para gestionar los activos de conocimiento (en esta tesis doctoral, los patrones de proyecto) deben gestionarse atendiendo a su ciclo de vida. Su gestión pasa por las siguientes fases:

- Identificación: El conocimiento es creado en la organización, a través de sus miembros por medio del aprendizaje, resolución de dificultades, cambio de procedimientos y experimentación de modelos externos.
- Captura: El conocimiento es capturado por la organización y sus miembros siguiendo un proceso de formato explicito con sus componentes.
- Organización: El conocimiento capturado se almacena, se clasifica y se organiza para un mejor control y uso.
- Distribución: El conocimiento es distribuido a los miembros de la organización que lo necesitan a través de técnicas de información.
- Preservación: El conocimiento se procesara y mejorara de forma continua y permanente, admitiendo los cambios favorables que propongan sus miembros.
- Uso: El conocimiento almacenado se aplica en el desarrollo y mantenimiento del software para los procesos de funcionamiento definidos en la organización.
- Medición: El conocimiento se debe evaluar para poder comprobar si la consecución de objetivos fijados por la organización a través de las actividades de gestión del conocimiento adquirido. Las organizaciones de desarrollo de software deben ser el soporte de desarrolladores y la creación de software de máxima calidad.

El control y medición del conocimiento permite identificar conocimiento nuevo, así el modelo propuesto es parecido al modelo en espiral que refleja la naturaleza iterativa de como gestionar el conocimiento. Podemos ver una descripción gráfica del modelo en la Figura 5

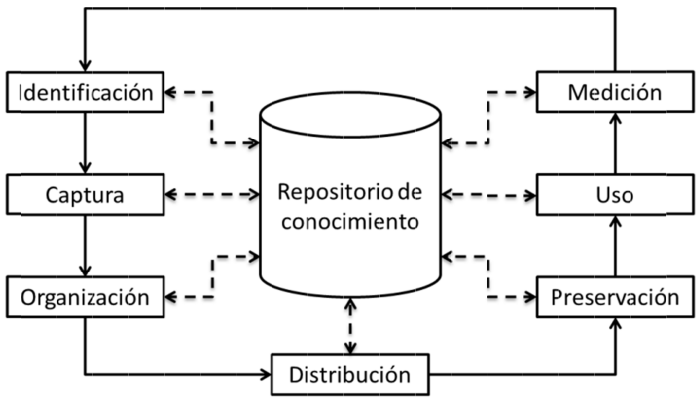


Figura 5 Modelo de gestión del conocimiento de Rus (Rus & Lindvall, 2002)

La mayor parte de los proyectos de gestión del conocimiento crean repositorios de conocimiento, mejoran la forma de acceso al conocimiento o tratan de mejorar el aspecto socio-cultural y un mejor sistema de intercambio de conocimiento.

La base de un buen sistema de gestión del conocimiento esta en el repositorio (llamado también memoria corporativa o memoria organizativa) que define el reuso y comparte no solo el conocimiento de la organización y las lecciones aprendidas con anterioridad. La idea principal del repositorio es contener los mecanismos de conocimiento (activos de proceso de software) en instrumentos utilizables y recuperados para ser utilizados en necesidades futuras.

La codificación del conocimiento requiere que el conocimiento se capture, codifique, organice y almacene en un formato apropiado. Entre los objetivos de un repositorio de conocimiento se encuentran (Stoyko et al., 2007):

- Identificar y proteger el conocimiento estratégico de la organización.
- Compartir y estimular el conocimiento entre los miembros de la organización.
- Mejorar la retención de trabajadores y prevenir la pérdida de conocimiento atribuible a la salida de trabajadores.
- Integrar el conocimiento dentro de la empresa.

Las formas y funciones de los repositorios de conocimiento pueden ser vistas como concretas o abstractas almacenando conocimiento de dos tipos (Jennex, 2004): Información precisa y estructurada en bases de datos, archivos y mecanismos, y la representación de información abstracta no estructurada de miembros de la organización. Los diferentes tipos de repositorios de conocimiento y sus categorías es dada por (Mentzas et al., 2001):

- Repositorios de Conocimiento Externo: Con ítems de información disponibles desde artículos y reportes para ejecutivos hasta sistemas avanzados de inteligencia de clientes.
- Repositorios Internos Estructurados: Incluye información como reportes de investigación, materiales de mercadeo orientados al producto, y técnicas y métodos.
- Repositorios Internos Informales: Con información como lecciones aprendidas, con conocimiento que debe ser interpretado y adaptado por el usuario en un nuevo contexto.

2.3 Objeto de conocimiento y patrones

2.3.1 Definición de objeto de conocimiento

Según (Merrill, 2000) la psicología cognitiva sugiere que el modelo mental consta de dos componentes principales: estructuras de conocimiento (o esquemas) y los procesos para el uso de este conocimiento (operaciones mentales). Una de las principales preocupaciones del diseño de la instrucción es la representación y organización de los contenidos subjetivos de manera que faciliten el aprendizaje. En el trabajo de (Merrill, 2000) se sugiere que el análisis del conocimiento puede facilitar tanto la representación exterior del conocimiento como fines instructivos (objetos de conocimiento) y la representación interna y la utilización de los conocimientos por los alumnos (modelos mentales). Los diseñadores de este conocimiento han reconocido la importancia de analizar el conocimiento a través de una selección apropiada del conocimiento, organización y secuencia. Mientras que los instructores tienden a centrarse en los sistemas de entrega (especialmente la tecnología) y en menor medida en las estrategias y tácticas de instrucción.

En la investigación llevada a cabo por (Merrill, 2000) propone un esquema de representación de conocimiento que consiste en componentes de conocimiento organizados en objetos de conocimiento, en dicho trabajo se sugiriere que casi todo conocimiento cognitivo podría ser representado como cuatro tipos de objetos de conocimiento. Las entidades son las cosas (objetos), las acciones son los procedimientos que puede ser realizado por los aprendices. Los procesos son los eventos que ocurren a menudo como resultado de una acción y las propiedades son descriptores cuantitativos o cualitativos para las entidades, acciones y procesos. En el trabajo de Merrill se definió el conocimiento a través de la composición de objetos de conocimiento. Un objeto de conocimiento y sus componentes son una forma precisa de describir el contenido para ser enseñado. Las composiciones de objetos de conocimiento son un conjunto de contenedores definidos de información.

2.3.2 Patrones

En esta sección se va a realizar un análisis de los patrones más difundidos en la literatura, este análisis se realizará siempre desde el punto de vista de su utilidad para la gestión del conocimiento referente a patrones de proyecto. Además trataremos a estos patrones como objetos de conocimiento.

Existen muchas definiciones de patrones en la literatura y en la Ingeniería del Software, pero la inicial fue dada por el arquitecto Christopher Alexander en 1979 como un concepto arquitectural:

"Un patrón describe un problema que ocurre una y otra vez en nuestro entorno, describiendo el núcleo de la solución a dicho problema de tal forma que se puede usar esta solución un millón de veces sin hacerlo dos veces de la misma forma".
(Alexander, 1979)

2.3.2.1 Descripción de los modelos de patrones más utilizados

La estructura de un patrón se define como el modelo de datos o el formato de un patrón. Un modelo de datos ofrece la estructura de la información. Existen varios modelos de patrones (Fowler, 2003). Los modelos más extendidos y aceptados por la mayoría son: el modelo de Alexander (Alexander et al., 1978), el modelo GoF (Gamma, 1995) y el modelo canónico (Buschmann, 2007).

2.3.2.2 Modelo Alexander

Este modelo expone estilo narrativo del problema que se quiere resolver, las condiciones aplicables y la solución propuesta. El esquema típico usado por Alexander lo integran los elementos siguientes:

- Nombre
- Problema
- Contexto
- Restricciones
- Solución

2.3.2.3 Modelo GoF

Modelo propuesto y desarrollado por los autores Erich Gamma, Richard Helm, Rala Johsson y John Vlissides, conocidos como el "Gang of Four" o "Club de los Cuatro" los patrones están descritos y catalogados en su libro (Gamma, 1995). La estructura y el formato de la plantilla se describen en la Tabla 1 Plantilla modelo GoF

SECCIÓN	DESCRIPCIÓN
Nombre del patrón y clasificación	Expresa la esencia del patrón. Se define su propósito y ámbito.
Propósito	Breve introducción que responde a las preguntas: ¿Qué hace el patrón? ¿Cuál es su razón? ¿Qué problema resuelve?
Otros nombres	Otros nombres dados al patrón.
Motivación	Escenario que ilustra el problema.
Aplicabilidad	Situaciones en las cuales el patrón puede ser aplicado. Pautas para reconocerlas.
Estructura	Representación gráfica de las clases que conforman el patrón. Diagrama de clases de colaboración, etc.
Participantes	Clases y/u objetos que participan en el patrón y sus responsabilidades.
Colaboraciones	La forma en la cual los participantes colaboran para realizar el cometido.
Consecuencias	¿Cómo soporta el patrón sus objetivos? ¿Cuáles son sus concesiones?
Implementación	Trucos o técnicas que deben considerarse en la implementación.
Código ejemplo	Fragmentos de código que ilustran la implementación del patrón.
Patrones relacionados	Enumeración de otros patrones relacionados con el que se describe. Diferencias entre los mismos.

Tabla 1 Plantilla modelo GoF

2.3.2.4 Formato Canónico

Definido por Buschmann (Buschmann et al., 2007) para la descripción y desarrollo de patrones. Esencialmente, desarrolla los mismos apartados que el formato Alexander (con frecuencia se utilizan indistintamente los dos sistemas de forma complementaria) aunque con mayor detalle y precisión este ultimo. Se le conoce también como modelo POSA (Tabla 2 Plantilla modelo canónico).

SECCIÓN	DESCRIPCIÓN
Nombre	Nombre significativo. Puede ser una única palabra o una frase corta que haga referencia al patrón.
Problema	Descripción del problema indicando su propósito.
Contexto	Precondiciones bajo las cuales se puede aplicar el patrón.
Restricciones	Descripción de las restricciones relevantes y cómo interaccionan o entran en conflicto unas con las otras. Definición de los objetivos a conseguir.
Solución	Relaciones estáticas y reglas dinámicas que describen cómo conseguir el objetivo marcado.
Ejemplos	Uno o más ejemplos de la aplicación del patrón.
Contexto resultante	Estado o configuración del sistema después de haber aplicado el patrón. Describe tanto los efectos positivos como perniciosos.
Razón	Explicación justificada de los pasos o reglas del patrón.
Patrones relacionados	Relaciones estáticas o dinámicas entre el patrón que se describe y otros.
Usos conocidos	Descripción de ocurrencias conocidas del patrón y su aplicación dentro de sistemas existentes.

Tabla 2 Plantilla modelo canónico

2.3.3 Tipos de patrones de software

Existen varias clasificaciones de los tipos de patrones siendo múltiples los trabajos y estudios habiendo aumentado enormemente en los últimos años. En la Figura 6 se muestra una estructuración de los diferentes patrones en la ingeniería del software.

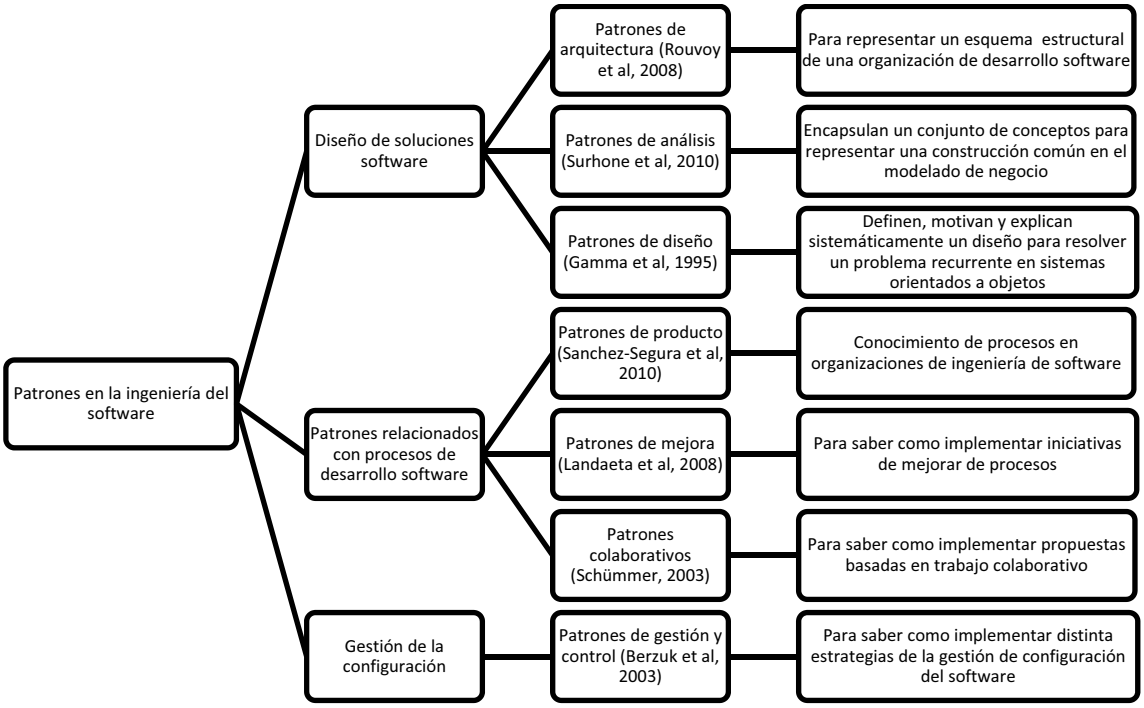


Figura 6 Clasificación de los patrones

En la ingeniería de software se detallan ocho tipos de patrones en tres niveles de abstracción y dominios de conocimiento diferentes, como se ha visto en la Figura 6.

2.3.3.1 Patrones de implementación

Subdivididos en 4 tipos dependiendo el dominio de aplicación.

a) Arquitecturas de referencia. Arquitectura que ya ha sido pensada y creada para la utilización de un dominio de interés particular y concreto. Suele tener diversos estilos arquitectónicos, aplicados en diferentes partes de su estructura. Podemos encontrar varios ejemplos de este tipo de patrones: Dos capas (Two tier), Multicapa (Multi-tier), Workflow, Basada en Documentos (ej. Notes), Orientada a Formularios, Orientada a Hipertexto, Video, Imagen, Multimedia, etc.

b) Patrones de Arquitectura. Define el un esquema estructural y esencial de una organización en los sistemas de software., define sus responsabilidades, proporciona una variedad de subsistemas predefinidos e interrelaciona y organiza dichos sistemas a través de normativa reglada. Ejemplos: El patrón Model-View-Controller (MVC).

c) Patrones de Análisis. Fue definido por Fowler en 1997 (Fowler, 1997) como el ensamblaje común de un modelo de negocio, agrupando distintas concepciones de agrupamiento. Su relevancia para un solo conjunto de problemas o para varios sistemas. Son los más utilizados por los desarrolladores. Ejemplos: Party, Quantity, etc.

d) Patrones de diseño. Explica y define sistemáticamente un diseño genérico para la solución de problemas recurrentes de diseño en sistemas dirigidos y orientados a objetos. Analiza el problema, da las pautas de solución, fija cuando se debe aplicar y define las posibles consecuencias. La solución es un modelo ordenado de objetos y clasificaciones que da solución a las cuestiones planteadas, resolviendo el problema en su conjunto y en su contexto dentro de la empresa con una dinámica de construcción e implementación del sistema propio (Gamma, 1995). Ejemplos: Adapter, Factory Method, Composite, Iterator, etc.

2.3.3.2 Patrones de Proceso

Un patrón de proceso señala y analiza una situación probada con éxito anteriormente en múltiples de acciones de desarrollo software (Ambler, 1998) (Ambler, 1999). Ambler, en ambos trabajos, concreta en patrones de proceso basados en la base orientada a objetos y propone tres niveles de patrones; 1) Patrones de proceso de tarea, 2) Patrones de proceso de nivel y 3) Patrones de proceso de fase. La unión de estos tres niveles cumple con el objetivo de dar al desarrollador la herramienta necesaria en los diferentes niveles a la hora de llevar a cabo un desarrollo software orientado a objetos.

En (Iida, 1999) se establece el concepto de evolución de los patrones de proceso y se introduce en su aplicación. En (Gnatz et al., 2003) se propone un framework para relacionar y describir los procesos de desarrollo software modulado en base a patrones de proceso. En este trabajo se analiza una aplicación web llamada LiSA que ayuda a definir y concretar los procesos de desarrollo software de las organizaciones para un mejor mantenimiento del mismo. Dichos patrones de proceso tienen su base en una plantilla denominada "Plantilla para Patrones de Proceso de Desarrollo Software".

2.3.3.3 Patrones de Mejora de proceso software

Los patrones de mejora de procesos software (Landaeta, 2008) hacen que la calidad del sistema implementado en la organización se perfeccione y mejore aplicando las buenas prácticas. (Appleton, 1997). Estos patrones son definidos y

estructurados: nombre, contexto, problema, restricciones, solución, contexto resultante, base lógica, patrones relacionados y usos conocidos. Los patrones de mejora de procesos han sido determinados y clasificados por Appleton en dos categorías destacables: patrones organizacionales que están enfocados por propia definición a la organización de la empresa, y patrones de proceso y comunicación encaminados a la eficiencia e interconexión de sus miembros.

2.3.3.4 Patrones de Gestión y Control

Los patrones más importantes en esta rama concreta son los patrones de gestión de la configuración, donde en (Berczuk, 2003) se puede percibir una posible definición del lenguaje de patrones que permita su aplicación de forma muy concreta al proceso de la Gestión de Configuración del Software (SCM). Los patrones señalados y analizados permiten acceder a unas heurísticas que lleven a cabo las tareas de SCM. Además señala y marcan los instrumentos necesarios para soportar la ejecución de dichos patrones. Todo esto no es un tema de descripción formal de los mismos, ni tan solo la señalización y anotación de los campos de estructuración del patrón se han catalogado como campos identificativos que distribuyan y estructuren dichos patrones aunque sirvan también para el almacenamiento y procesado de la información puntual sobre la que ejecutar los patrones de SCM.

2.3.3.5 Patrones organizativos

Según (Ambler, 1999) un patrón organizativo es aquel que describe una técnica de gestión organizativa por capas o un potencial estructural organizado del proceso. En (Isla-Montes, 2004) los patrones organizativos son llevados a la practica para la modelación de la estructura y el engranaje social del factor humano que interviene en el sistema dentro de una pedagogía de desarrollo programático. Proponen el modelo de tres capas estructuras de la propia organización en patrones de desarrollo organizativos. Al poner en práctica la utilización de los referidos patrones consiguieron la validación de los patrones al facilitar, no sólo el diseño específico del sistema, si no también la actualización y modelización de los mismos. Queda patente las múltiples ventajas aportadas a la organización al utilizar un patrón organizativo como premisa básica de funcionamiento aunque dicho patrón rompa con el nivel conceptual predeterminado.

2.3.3.6 Patrones colaborativos

En esta rama de patrones colaborativos se han logrado catalogar e identificar diez patrones (Sommerville, 2004), visibles en web, pero no

necesariamente en relación directa con la ingeniería del software. En (Campbell, 2004), se propone un diseño, CoDiagram, que permite utilizar los elementos integrantes a través de diseños de colaboración. Dicha herramienta se apoya sobre Visio y suministra numerosas alternativas de funcionamiento que puedan colaborar en el soporte propio del comportamiento en equipo. En (Lukosch & Schümmer, 2006) usan los patrones colaborativos para la puesta en marcha de plataformas groupware. En la Tabla 3 Modelo del Patrón Colaborativo (Lukosch, 2004), se muestra la estructuración del patrón colaborativo propuesta y definida por Lukosch (Lukosch & Schümmer, 2004).

SECCIÓN	DESCRIPCIÓN
Nombre	Nombre del patrón colaborativo.
Propósito	Un resumen de la intención y lógica o heurísticas del patrón.
Contexto	Situación o estado de un proyecto en el cual el patrón de proceso pueda ser aplicable.
Problema	El elemento que dirige el patrón, incluyendo una discusión de restricciones asociadas proporcionando un escenario o ejemplo del mundo real.
Solución	Describe las mejoras proporcionadas después de haber aplicado el patrón.
Escenario	Descripción concreta de una situación donde el patrón puede ser utilizado para que se vea de forma más tangible el problema.
Síntomas	Ayuda a identificar la necesidad de los patrones describiendo aspectos más abstractos del patrón.
Colaboraciones	Indica los principales componentes o actores que interactúan con el patrón y explica cómo se relacionan entre sí.
Razón	Explica por qué las restricciones son resueltas por el patrón.
Peligros	Descripción de las restricciones en contra que se puede encontrar al aplicar el patrón colaborativo.
Usos conocidos	Descripción de ocurrencias conocidas del patrón y su aplicación dentro de sistemas existentes. Ejemplos que muestren la utilidad del patrón.
Patrones relacionados	Referencias a patrones que resuelvan problemas similares.

Tabla 3 Modelo del Patrón Colaborativo (Lukosch, 2004)

2.4 Soluciones tecnológicas para la gestión de conocimiento

En esta sección se presenta el estudio realizado a través de las soluciones tecnologías que dan soporte a la gestión del conocimiento. Como en el resto de las secciones se ha prestado una mayor atención a aquellas que pueden ayudar a la gestión del conocimiento referente a patrones de proyecto.

2.4.1 Tecnología para la gestión del conocimiento

En esta sección se presentarán las diferentes tecnologías que pueden servir para la gestión del conocimiento.

2.4.1.1 Repositorios de experiencia

En 1994 (Basili et al., 2002), crearon el concepto de "Experience Factory" para las organizaciones cuyo periodo de aprendizaje se basa en repositorios de experiencia. Posteriormente ha habido múltiples investigaciones dirigidas a implantar los repositorios de experiencia en la mecánica de actuación de las organizaciones (Dingsøyr, 2000; Feldmann & Carbon, 2003; Basili et al., 2007).

2.4.1.2 Wikis en ingeniería del software

En el año 2005 (Chau & Maurer, 2005) proponen que los repositorios de experiencia sean representados por tecnología Wiki (Cunningham & Leuf, 2001). Este estudio demuestra la gran aceptación de la tecnología Wiki como instrumento necesario para compartir el conocimiento en las organizaciones. También es de destacar la necesidad de que las organizaciones guarden el conocimiento no estructurado para su utilización futura.

En 2007, Petter y Vaishnavi (Petter & Vaishnavi, 2007) investigaron cómo solucionar el problema de la falta de experiencia que tenían los jefes de proyecto software para controlar y dirigir un proyecto de forma adecuada, este problema una de las causas que provocan que solo el 29% de 9000 proyectos de desarrollo software (Standish Group, 2004) finalicen en el tiempo establecido y con las característica de funcionamiento propuestas.

En su estudio, el objetivo fue volver a utilizar la experiencia obtenida por los jefes de proyectos en software anteriores. Para la solución de dicho problema, se propuso la creación un colectivo especializado en gestión de proyectos (Project Management Community) que dispusiera de bibliotecas de intercambio de experiencia (Experience Exchange Library).

2.4.1.3 Web semánticas.

El sistema basado en la reutilización semántica (SRS) para la adquisición y gestión del conocimiento de software, en el trabajo de (Antunes et al., 2007) se describe un sistema basado en sistemas de reutilización semánticas.

En este trabajo se proponen la diferenciación de tres capas lógicas:

- Capa de datos: Base de conocimiento integrada por una ontología de representativa, una ontología del dominio, un repositorio de los elementos integrantes del proceso de desarrollo software (Software Development Knowledge Element - SDKE) y los datos del sistema y del usuario. Los datos se representan con RDF, el RDF Schema (RDFS) y Web Ontology Language (OWL)
- Capa de núcleo: El sistema es controlado en las siguientes operaciones: indexación de los SDKE; el Módulo de Contexto que une el contexto de trabajo del usuario; y el Módulo de Búsqueda que utiliza búsquedas semánticas para localizar el conocimiento relevante.
- Capa de Interfaz: Integrada por un conjunto de herramientas para gestionar el sistema y el conocimiento básico.

2.4.2 Librerías de activos de proceso (PAL)

Las librerías de activos de proceso (PAL) se identifican como un repositorio organizado, bien indexado, de activos de procesamiento de datos de fácil acceso por cualquier miembro de la organización que necesite alguna clase de información del proceso en marcha, ejemplos pueden ser: documentos de datos, plantillas generales o especificas o cualquier otro material definido en los proyectos software (Sheard, 2003).

Las librerías de activos de proceso proporcionan a los miembros de la organización:

- Acceder a las descripciones de procesos.
- Entender todos los procesos a nivel general.
- Entender en detalle los procesos que cada persona realiza.

Para comprender el concepto de PAL, se realiza una descripción de sus elementos constituyentes: los activos de proceso. Luego, cómo dichos activos pueden ser almacenados y recuperados en una biblioteca digital e incluir funcionalidades adicionales para convertirse en una PAL. También se presentan ejemplos de utilización de estas librerías de activos.

2.4.2.1 Activos de proceso

Las descripciones de los procesos de la organización usualmente contienen una secuencia de pasos a ser ejecutados, identifican quiénes los ejecutan, especifican los criterios de entrada y salida para las principales actividades, etc. Para apoyar el uso de procesos, frecuentemente se proporciona guías, listas de chequeo y plantillas. Estos materiales son llamados activos de procesos (Jalote, 2002).

Los activos de proceso son cualquier elemento que la organización considere útil para realizar con éxito el proceso de desarrollo de un producto software. Pueden definirse como una colección de entidades mantenidas por una organización para su uso en proyectos durante el desarrollo, adaptación, mantenimiento e implementación de sus propios procesos (Paulk et al., 1993).

Los activos se convierten en artefactos que permiten describir, implementar y mejorar los procesos software definidos en la organización. El término "activo" se utiliza para resaltar que estos artefactos permiten cumplir los objetivos de negocio de la organización y que son "inversiones" con las cuales la organización espera alcanzar valores actuales y futuros (SEI, 2010) Además, los activos proveen el fundamento para institucionalizar el proceso en cualquier organización. En la Tabla 4 se presenta un listado con algunos activos de proceso típicos de una organización desarrolladora de software (Layman, 2005).

Activos	Descripción
Ciclos de vida	Periodos de tiempo consistentes en fases que comienzan y finalizan cuando el producto ya no está disponible para su uso, debido a que las organizaciones generalmente producen múltiples productos se tendrán varias descripciones de ciclos de vida aprobados para el desarrollo.
Procesos	Definiciones operacionales de los principales componentes de un proceso para la realización de proyectos junto con mediciones recolectadas durante la ejecución.
Guías y criterios de adaptación	Guías definidas en la organización que permiten a proyectos, grupos o funciones organizativas adaptar apropiadamente el proceso estándar para su utilización.
Plantillas	Estructuras básicas que proporcionan la estructura de documentos que pueden ser utilizados para crear productos de trabajo en las descripciones de proceso. Por ejemplo, plantillas para planes de proyecto, planes de pruebas, requisitos, documentos de diseño, etc. Las plantillas dan agilidad para crear y revisar productos de trabajo.
Guías	Información útil y detallada sobre cómo realizar una actividad. Las guías sirven como orientación y no son obligatorias. Son útiles cuando una actividad no es ejecutada frecuentemente y va a ser ejecutada pro primera vez, cuando hay múltiples formas de ejecutar un proceso o cuando hay varias técnicas que pueden utilizarse dependiendo de las condiciones establecidas.
Estándares	Reglas aplicables a un tipo particular de productos de trabajo. Por ejemplo, estándares de codificación de programas en un lenguaje de programación.
Listas de verificación	Ayudas para valorar la completitud o exactitud de una actividad o producto de trabajo. Típicamente tienen la forma de un conjunto de preguntas que sirven como indicadores a los usuarios o revisores del proceso.
Lecciones aprendidas	Buenos ejemplos tomados de proyectos pasados que pueden ser activos útiles para la realización de proyectos futuros.
Material de entrenamiento	Materiales de referencia para entrenamiento formal sobre procesos, herramientas o métodos que pueden ser activos valiosos.
Mediciones	Mediciones sobre la ejecución y utilización de los procesos con el propósito de caracterizar y entender el proceso.
Otros	Cualquier documentación relacionada con el proceso software

Tabla 4 Lista de activos de proceso

2.4.2.2 Guías Electrónicas de Proceso

En 1998, (Kellner et al., 1998) definieron la guía electrónica de proceso (EPG) como un documento base orientado al intercambio de trabajo con estructura propia para llevar a término la ejecución de un proceso. La guía electrónica de proceso señala y concreta los campos del proceso: sus acciones, mecanismos integrantes, roles y recursos que directamente están relacionados en el proceso y la relación entre componentes.

Las guías electrónicas de proceso son aplicadas en puntos diferenciados de investigación. Una de las líneas de investigación es el avance y la actualización del proceso software, donde se encuentra el trabajo de (Moe & Dybå, 2006). En dicho estudio se llevo a cabo una encuesta a las personas participes en un proyecto software para evaluar la forma en que les llegaba la información y como percibían las características de control y mejora en las guías electrónicas de proceso (la eficacia en de uso, el aprendizaje del proceso y su utilidad). Los resultados a los que llegaron en dicho estudio se concreto en que las guías electrónicas de proceso general hay que adaptarlas para su uso concreto, a los procesos propios y específicos de la organización para que sean útiles a quienes las utilizan. Sobre la facilidad de su uso, los resultados obtenidos fueron variados, dependiendo de las características de la organización y del nivel de sus componentes, para los jefes de proyecto la EPG era fácil de utilizar, en cambio para los desarrolladores, la complicación del desarrollo diverso genera una gran dificultad de utilización y de complementación en el sistema guía. Las conclusiones finales se pueden concretar en que la guía electrónica de proceso está dirigida a los procesos y no al producto final y tienen que ser definidas y adaptadas a las peculiaridades propias de la empresa.

2.4.2.3 Activos de proyecto

Son todos los productos y documentación técnica que se produce después de la ejecución de las actividades del proceso de desarrollo software.

2.4.3 Herramientas que soportan la gestión de conocimiento

El conocimientos sobre procesos del software normalmente se gestiona en repositorios llamados “Process Asset Libraries” (PAL). Estos repositorios ofrecen las funciones de almacenar, gestionar, distribuir y reusar los activos de procesos para poder recopilar todos los casos y experiencias de éxito, contribuyendo a la mejora del proceso software (Garcia & Turner, 2006).

En esta sección se discutirán las principales herramientas que dan soporte a la gestión del conocimiento.

Además de las presentadas en esta sección merece la pena destacar herramientas presentadas en las tesis doctorales (Medina-Dominguez, 2010), donde se presenta un marco metodológico para la mejora del proceso de desarrollos software y en (Bermón, 2010) donde se muestra una herramienta para gestionar librerías de activos de procesos (PAL) usando la tecnología Wiki (Cunningham, 2001)

2.4.3.1 EzyLib

EzyLib es una herramienta para crear una PAL que se centra en la definición de una estructura flexible para organizar los activos de proceso. La herramienta ofrece las siguientes funcionalidades (Group Processworks, 2011):

- Navegación sobre la estructura de la PAL de acuerdo a múltiples vistas.
- Visibilidad de la información acerca de los activos. Por ejemplo, visibilidad por roles aplicables, materiales de referencia, ejemplos y/o revisiones históricas.
- Búsqueda de activos basada en palabras claves, popularidad o relevancia.
- Facilidades para la gestión de la PAL y recolección de métricas sobre su uso.

2.4.3.2 EPF Composer

EPF Composer (The Eclipse Foundation, 2011) es una herramienta para el entorno de desarrollo Eclipse que utiliza un enfoque basado en formularios para definir los contenidos de los procesos.

La herramienta proporciona una infraestructura extensible para crear, configurar y publicar procesos. Proporciona a los desarrolladores una base de conocimiento que permite navegar, gestionar y presentar contenidos como definiciones de métodos, guías, plantillas, mejores prácticas, material de entrenamiento y descripciones generales sobre cómo desarrollar software, ver Figura 7

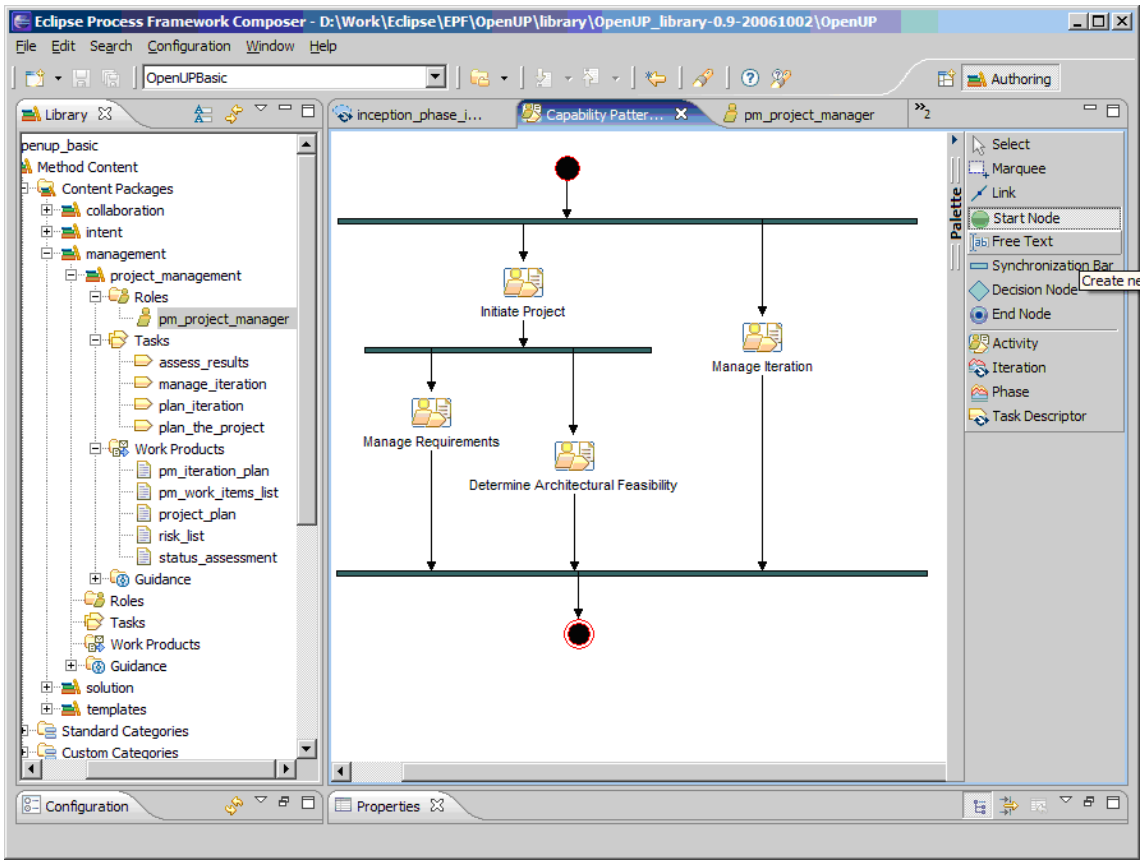


Figura 7 Captura de pantalla de EPF Composer.

La herramienta también proporciona capacidades de ingeniería de procesos dando soporte a los gestores de proyectos en seleccionar, adaptar y ensamblar procesos para proyectos concretos mediante bloques de construcción. El proceso documentado creado puede ser publicado como un sitio Web.

La herramienta actualmente soporta tres marcos de trabajo de procesos: Proceso unificado, Programación extrema y Scrum. También se pueden crear marcos de trabajo personalizados desde cero. Esta herramienta ofrece algunas funcionalidades como: seleccionar y configurar procesos existentes, adaptar procesos existentes, crear nuevos procesos y desarrollar contenidos para los procesos.

La versión comercial de *EPF Composer* se denomina *Rational Method Composer* y forma parte de la suite de *Rational* proporcionada por *IBM*. Todas las características proporcionadas por *EPF Composer* están disponibles en esta versión comercial.

2.4.3.3 Select Process Director

Select Process Director consiste en varias herramientas que permiten la creación, modificación y el uso de elementos de un proceso almacenados en un repositorio como se observa en la Figura 8

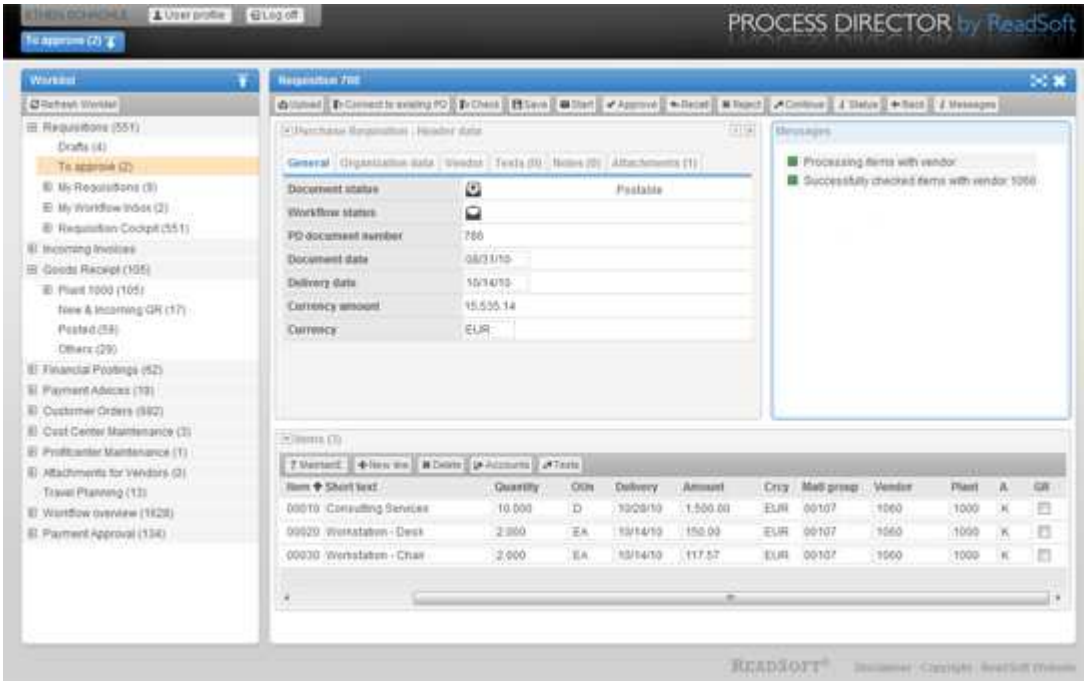


Figura 8 Captura de pantalla de Select Process Director

La herramienta *Select Process Director* ofrecen las siguientes funcionalidades orientadas a colocar el proceso de software en la práctica (Select Business Solutions, 2011):

- Definición y almacenamiento de los elementos del proceso en el repositorio.
- Construcción de un proceso real a partir de los elementos existentes en el repositorio. De esta forma, los administradores de proyectos pueden aplicar los procesos de acuerdo a sus necesidades.
- Actualización y monitorización en tiempo real de las actividades del proyecto conforme el proyecto avanza. Los miembros del equipo del proyecto puedan hacer referencia al trabajo diario en el proceso y reportar el estado y progreso de los proyectos.

2.4.3.4 IRIS Process Author

Iris Process Author es una herramienta comercial de la compañía Oselus (Osellus, 2007). Los procesos pueden ser modelados conforme a la especificación SPEM aunque sus componentes pueden ser personalizados. Los autores crean librerías de procesos para diferentes procesos y luego ingresan todos sus contenidos como roles, productos de trabajo, tareas, guías, etc. y los organizan en paquetes. Los paquetes pueden importar elementos de proceso desde otros paquetes y así crear paquetes personalizados.

Los paquetes pueden ser exportados o publicados en diferentes formatos como HTML, Microsoft Word, Microsoft Project y PDF

2.4.3.5 EssWork

Es una herramienta que implementa el enfoque basado en "Prácticas" desarrollado por Jacobson (Jacobson et al., 2007). Una práctica se define como: "Una forma probada de dirigirse a un problema. Es algo que se ha realizado antes, puede ser comunicado a los demás, y puede ser aplicado repetidamente obteniendo resultados consistentes".

Las prácticas pueden estar dirigidas a diferentes áreas del desarrollo de software como prácticas orientadas a la arquitectura, procesos, casos de uso, casos de uso de negocio, componentes, iteraciones, productos, equipos, proceso unificado, etc. como se observa en la Figura 9.

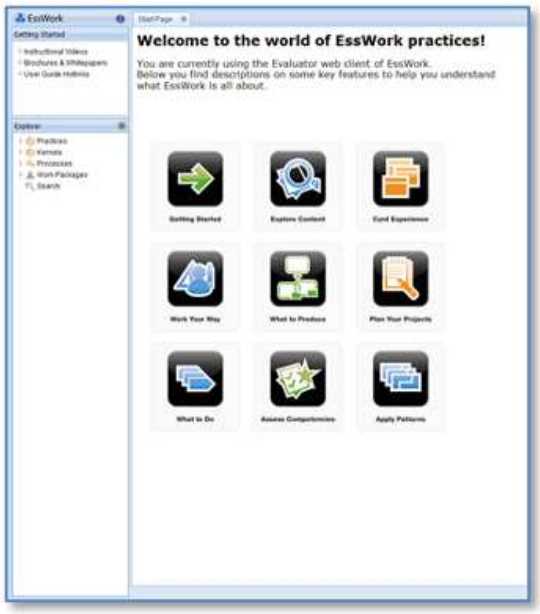


Figura 9 Captura de pantalla de EssWork

Los procesos se consideran como una colección de prácticas. Los desarrolladores seleccionan las prácticas que necesitan para un proyecto, pueden adaptar las que más les convienen o agregar nuevas prácticas a la herramienta.

La práctica sobre "procesos" mejora y adapta la forma de trabajo empleada por un equipo. Esta práctica permite:

- Identificar, preparar y ensamblar un conjunto de prácticas y herramientas para dar soporte a los objetivos de un proyecto.
- Introducir nuevas prácticas individual y gradualmente cuando sean necesarias.
- Evolucionar sus prácticas basadas en las experiencias y lecciones aprendidas.
- Cada práctica tiene asociada los siguientes elementos:
- Elementos a producir: Un conjunto de prácticas con su propia descripción en forma de tarjetas y guías, un conjunto de guías de herramientas que describen cómo ejecutar ciertas actividades definidas por las prácticas y un documento de resumen de la práctica.
- Competencias clave: Una descripción de las competencias para líderes, entrenadores y clientes.
- Elementos a hacer: Establecer y lanzar el conjunto inicial de prácticas y herramientas en el marco de la puesta en marcha de un proyecto.

2.4.3.6 Visual Studio 2010 Ultimate

Visual Studio 2010 Ultimate (Microsoft Corporation, 2010) es una solución que permite a un equipo de desarrollo colaborar y coordinar sus esfuerzos a la hora de crear un producto o llevar a cabo un proyecto, ver Figura 10

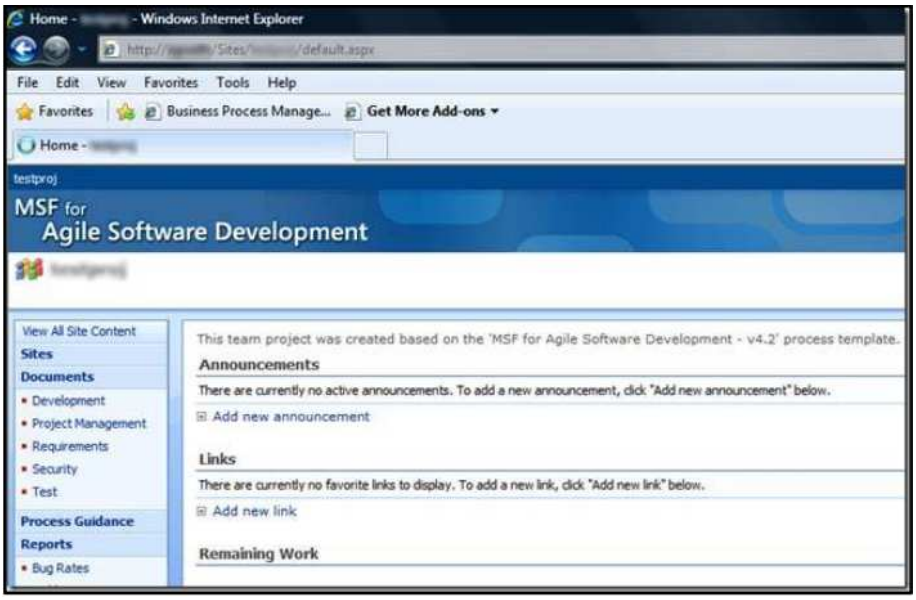


Figura 10 Captura de pantalla de Visual Studio 2010 Ultimate

Entre sus características se encuentran:

- Portal de proyecto: Cada proyecto de equipo tiene un portal asociado. Los miembros del equipo pueden utilizar el portal del proyecto para almacenar documentos, buscar informes y utilizar otras características de colaboración. Se proporcionan alertas que se envían a los miembros del equipo a través del correo electrónico cuando se realiza algún cambio en el proyecto.
- Seguimiento del estado del trabajo: Se vigila el estado de un proyecto, conociendo a quién se le ha asignado un trabajo y cuál es el estado de dicho trabajo.
- Representación del proceso: Los nuevos proyectos de equipo se crean a partir de una plantilla de procesos que define el conjunto de funciones en las que participarán todos los miembros del equipo. Los equipos pueden personalizar el proceso mediante un conjunto de ficheros XML.

2.4.3.7 Factoría de experiencia

La Factoría de Experiencia (Basili, 2002) es una organización lógica y/o física para analizar y sintetizar toda clase de experiencia, actuando como un repositorio, empaquetando la experiencia por medio de una construcción informal, formal o esquematizada de modelos y medidas de varios productos y procesos de software y otras formas de conocimiento a partir de documentos, personas o soporte automatizado Figura 11.

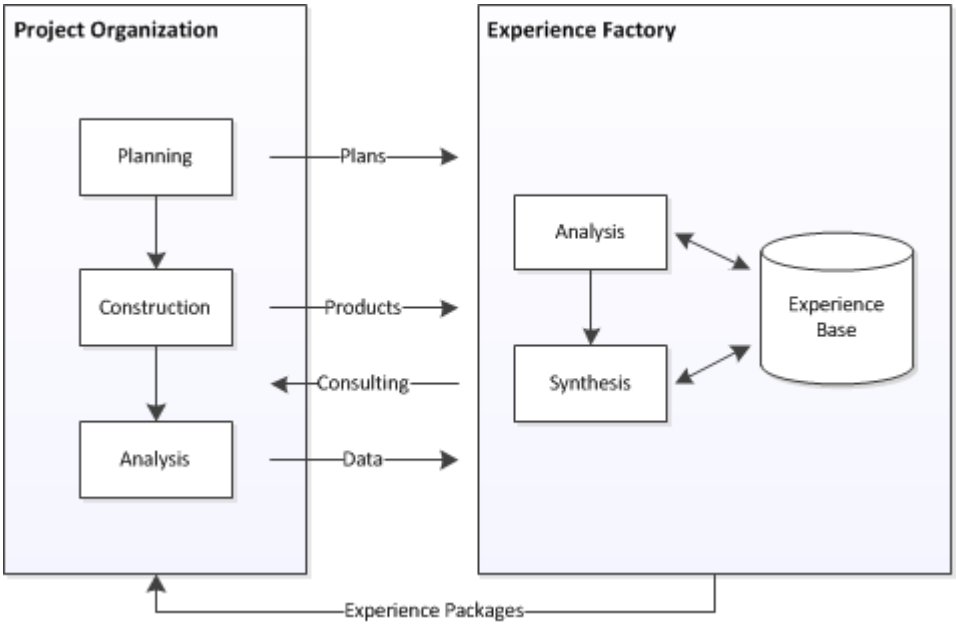


Figura 11 Factoría de experiencia

La factoría procesa esta información y retorna una realimentación directa de cada actividad del proyecto, junto con objetivos y modelos adaptados de proyectos previos. También produce, almacena y proporciona herramientas, lecciones aprendidas y datos desde una perspectiva más generalizada.

Entre los paquetes de experiencia que la factoría puede generar están los paquetes de productos, procesos, herramientas, de gestión, relaciones y datos. Los paquetes de procesos tienen como elemento central un proceso de ciclo de vida, junto con información necesaria para su ejecución y lecciones aprendidas durante su realización.

2.4.3.8 ProKnowHow

ProKnowHow es una herramienta basada en gestión del conocimiento para soportar la definición de procesos de software para un proyecto (Borges & Falbo, 2002).

Entre las funcionalidades de la herramienta, se encuentran:

- Soporte para la adaptación del proceso estándar a proyectos.
- Recolectar y diseminar el conocimiento adquirido durante la instanciación del proceso estándar.
- Soporte a la actualización del proceso estándar basada en la realimentación de proyectos.

El conocimiento acerca del proceso es almacenado en una memoria organizacional que contiene tanto conocimiento formal como informal. En el contexto de definición del proceso, hay dos clases de conocimiento formal: activos de proceso de software y artefactos para la definición del proceso.

2.4.3.9 Milos

Milos es una herramienta que integra la gestión del conocimiento y soporte al proceso de software para mejorar la eficiencia de equipos virtuales (Maurer & Holz, 2002). Los conceptos fundamentales son:

- Modelos genéricos de procesos con descripciones reutilizables de los procesos de desarrollo describiendo tareas, diferentes formas de resolver una tarea específica e información sobre el flujo de las tareas. Los modelos se almacenan en un Centro de Experiencias.
- Un Asistente de Información, que entrega información explícita acerca del proceso, basado en consultas automáticas a las fuentes de información asociadas a los modelos de procesos.
- Un enfoque de aprendizaje organizacional que permite empaquetar elementos "buenos" de proyectos exitosos soportando estrategias de mejora continua.

2.4.3.10 EPG/ER

EPG/ER es una guía electrónica de procesos junto con un repositorio de experiencias (Scott et al., 2002). La guía electrónica está formada por un conjunto de páginas HTML que poseen enlaces hacia una interfaz al repositorio de experiencias. Este repositorio almacena elementos como listas de verificación, plantillas, ejemplos y experiencias no estructuradas que pueden ser ingresadas o accedidas por los usuarios a través de la Web.

2.4.3.11 Resumen

En esta sección se va a realizar un resumen del aporte de cada una de las herramientas analizadas desde el punto de vista del ciclo de vida de los patrones de proyecto. Se analizaran que tareas de los ciclos de vida son soportadas por dichas herramientas. La Tabla 5 ofrece un resumen del análisis realizado.

	Adquisición	Organización	Distribución	Uso	Conservación
EZyLib (Group Processworks, 2011)	X	X	X	X	X
EPF Composer (The Eclipse Foundation, 2011)	X	X		X	
Select Process Director (Select Business Solutions, 2011)	X	X		X	X
IRIS Process Author (Osellus, 2007)	X	X	X	X	X
EssWork (Jacobson, 2007)	X	X		X	
Visual Studio 2010 Ultimate (Microsoft Corporation, 2010)	X		X	X	
Experience Factory (Ivarsson & Gorschek, 2011)	X	X	X	X	
Knowledge Packages (Ardimento et al., 2009)	X	X	X	X	X
Milos (Maurer, 2002)	X			X	
EPG/ER (Scott et al., 2002)	X			X	
ProKnowHow Tool (Borges, 2002)	X	X	X	X	X

Tabla 5 Herramientas para el soporte de la gestión del conocimiento de procesos de desarrollo software

La Tabla 5 muestra las plataformas tecnológicas actuales que dan soporte a la gestión de conocimiento de procesos de desarrollo software, se marcará con una X cuando la herramienta ofrece cobertura en la actividad concreta, y un blanco cuando la herramienta no ofrece cobertura para dicha actividad.

Durante la adquisición de conocimiento, éste es adquirido a través de la identificación de información relevante siempre siguiendo un modelo definido por cada herramienta software (García et al., 2011). El objetivo principal de la fase de adquisición es establecer los mecanismos efectivos que integren el conocimiento explicito y tácito necesario para componer un objeto de conocimiento válido (Amescua et al., 2010)

En la fase de organización, el conocimiento después de formalizarse debe ser clasificado adecuadamente para facilitar la indexación efectiva de los

elementos de información incluidos en un patrón de conocimiento. Existen varias plataformas para organizar activos de procesos en prácticas (Osellus, 2007), (Jacobson, 2007), (Basili, 2002), (Visaggio, 2009) y (Scott et al., 2002). En otros casos, los patrones son agrupados por procesos organizacionales, técnicas específicas de desarrollo software o ciclos de vida (The Eclipse Foundation, 2011), (Select Business Solutions, 2011), (Microsoft Corporation, 2010), (Maurer, 2002) y (Borges, 2002). Las plataformas más complejas de gestión del conocimiento proporcional un nivel adicional usando diferentes categorías para los procesos de software. El objetivo principal para este tipo de organizaciones es ayudar a los ingenieros de software a navegar a través de los diferentes patrones de proceso para encontrar el más apropiado para un proyecto concreto.

Una clasificación apropiada y un mecanismo de indexado es esencial para facilitar la distribución del conocimiento. La correcta clasificación y localización de los activos de proceso en la categoría adecuada es fundamental para mejorar el aprendizaje de los ingenieros sobre las prácticas requeridas en una organización de software. De esta manera, es importante que las plataformas de gestión del conocimiento faciliten estas tareas.

Un factor importante para poder mejorar la distribución de los patrones de procesos es la posibilidad de integrar y enriquecer otros objetos de conocimiento independientemente de la plataforma tecnológica utilidad para crearlo. Las plataformas actuales no consideran la posibilidad de buscar por un procesos disponible en Internet, pero creados con otras herramientas. De esta manera se necesitas definir soluciones tecnológicas para ayudar a la estandarización del área de patrones de proceso, mejorando el intercambio entre plataformas y usando técnicas de indización y búsqueda avanzadas que proporcionan Lucen (Apache Jakarta, 2005), Lemur (Allan et al., 2003) o RSHP (Llorens et al., 2004).

Durante el uso y la preservación de los patrones de procesos es necesario que se ofrezca a los ingenieros de software mecanismos que les ayuden a incluir activos de conocimiento procedentes de su propia experiencia durante el uso de los patrones en procesos de desarrollo software. Muchas organizaciones han implantado distintas plataformas para capturar el conocimiento individual de los ingenieros de software y distribuirlo entre el personal de la organización para poder constituir un repositorio de conocimiento de la organización (Jalote, 2002), (Basili, 2002) y (Visaggio, 2009). Las factorías de experiencia de (Basili, 2002) gestionan el conocimiento de procesos de desarrollo software de manera informal sin restricciones de representación de información relevante siguiendo un modelo de datos.

La preservación de los objetos de conocimiento depende del intercambio de conocimiento entre los ingenieros de software. En este sentido, la gestión del conocimiento se convierte un habilitador del aprendizaje organizacional (Aurum et al., 2008). A través de la interactuación y la compartición de conocimiento tácito y explícito con otros, los individuos mejoran la capacidad de adaptar nuevos procesos a las condiciones específicas y al contexto del proyecto de software (Nonaka, 2006).

Como conclusión a esta sección se puede afirmar que los patrones de proceso son artefactos efectivos para implementar estrategias de gestión del conocimiento que soporten iniciativas de mejora del proceso del desarrollo. Además, existen frameworks y herramientas para gestionar los patrones de proceso, incluyendo las fases de elicitación y de organización. Herramientas que soporten el ciclo de vida de los patrones de proceso

2.5 Análisis crítico al estado del arte

Como conclusión al análisis del estado del arte, se puede decir que los patrones de proceso son artefactos a ser considerados para la implementación de estrategias de gestión de conocimiento para soportar la mejora del proceso software. Por otra parte hay plataformas y herramientas para soportar la gestión del conocimiento como patrones de proceso en las fases de elicitación y organización. Sin embargo, es importante mencionar que es necesario realizar más investigación para analizar más en detalle para subsanar los problemas y deficiencias que se ha encontrado en la literatura. Se van a presentar estas deficiencias siguiendo la estructura que se ha utilizado para el análisis del estado del arte.

2.5.1 Análisis crítico de ciclos de vida para la gestión del conocimiento

Las estrategias actuales para la gestión del conocimiento sugieren varias fases o etapas pero no existe una específica que ofrezca cobertura total al conocimiento basado en patrones de proyecto.

Otro de los aspectos que requieren más investigación es que en la literatura analizada no se profundiza en los factores que ayudan a realizar de forma efectiva y eficiente cada una de estas etapas. En los trabajos de investigación analizados ha sido difícil encontrar estudios que nos aclaren los siguientes puntos:

- Los factores que influyen en la correcta elicitación y formación de los patrones de proyecto.

- Cuáles son los elementos de información más relevantes y apropiados para un proyecto software desde el punto de vista de las fases del ciclo de vida de patrones de proyecto.
- Cuál es la mejora en la calidad de los productos software si se utilizan las propuestas analizadas en el estado del arte.
- Cuál es el sobrecoste que supone el uso de las propuestas analizadas en el estado del arte.

2.5.2 Análisis crítico de los objetos para la gestión del conocimiento

Tras el análisis de la literatura, no se ha encontrado ningún artefacto que cubra completamente las necesidades de conocimiento de las organizaciones de desarrollo software al. Se han encontrado muchas propuestas pero muy pocas de ellas con un modelo claro y bien definido. Y de las propuestas que ofrecen un modelo, no ofrecen los ítems necesarios para cubrir las necesidades de las organizaciones de software.

En la literatura analizada no se han encontrado estudios que analicen los siguientes puntos:

- La utilidad de cada uno de los elementos de información incluidos en un patrón para adaptarlo a un proyecto software específico.
- Qué problemas podemos encontrar en ciertos elementos de información y como atajarlos para poder realizar implementar practicas eficientes en organizaciones de software.

2.5.3 Análisis crítico de las herramientas para la gestión del conocimiento

Se ha realizado un análisis del estado de la cuestión sobre las herramientas que ayudan a la gestión de artefactos y estrategias para la gestión del conocimiento.

En la literatura analizada es difícil encontrar herramientas que gestionen el conocimiento referente a procesos de desarrollo software tratados desde el punto de vista de un patrón de proyecto. Donde cada patrón es un objeto de conocimiento que puede ser adquirido, adaptado, utilizado, medido, etc.

Además, sería deseable que las herramientas analizadas potenciaran las siguientes características que serían muy deseables para un sistema de gestión del conocimiento:

- Cobertura total para todas las fases de las metodologías de conocimiento más utilizadas.

- Como hemos dicho antes, capacidad de modelar la información más relevante de las principales metodologías de desarrollo, buenas prácticas, marcos de referencia y el conocimiento tácito de las organizaciones.
- Un sistema de indexado de información para que puedan realizarse búsquedas efectivas sobre la información. El sistema de indexado que permita la búsqueda de información utilizando el lenguaje natural, búsquedas gráficas de elementos visuales, búsquedas a través de metadatos, búsquedas a través de ejemplos, etc.
- Un framework para la mejora del proceso de desarrollo software que integrara un modelo de datos, unos procesos que de soporte al conocimiento almacenado y una herramienta capaz de gestionar el modelo de datos a través de cada una de las actividades de la gestión del conocimiento.

SOLUCIÓN PROPUESTA

Tabla de contenidos: Solución propuesta

3: SOLUCIÓN PROPUESTA

Tras la investigación realizada en esta parte de la presente tesis doctoral se han publicado las siguientes contribuciones en conferencias: (Martinez et al., 2005), (Amescua et al., 2005), (Martín et al., 2007), (Martín et al., 2011a).

Este capítulo describirá de forma detallada el marco metodológico para la gestión del conocimiento de patrones de proyectos propuesto en esta tesis doctoral. En este capítulo además se detallará los objetivos, alcance, enfoque, los procesos, modelo sdPP y la herramienta para su gestión.

Este trabajo de investigación se enmarca dentro del proyecto de investigación "*Plataforma de gestión de procesos software: Modelado, reutilización y medición*.", cuyo investigador principal es Paloma Martínez Fernández; financiado por el Ministerio de educación y ciencia de España (Código 2004/02925/00).

3.1 Descripción y alcance

3.1.1 Descripción general

En esta tesis doctoral se ha optado por proponer un framework integral para dar cobertura a los problemas que se pretenden resolver en la misma, dicho framework se ha denominado sdpFramework; el cual esta compuesto por:

- Un modelo de conocimiento que se utilizará en las distintas fases del ciclo de vida del conocimiento de los patrones de proyecto llamado sdPP (Software Development Project Pattern);
- Un conjunto de procesos que describan las fases del ciclo de vida del conocimiento de patrones de proyecto llamado sdpProcessModel;
- Una herramienta, llamada sdpReuser (Software Development Pattern Reuser), capaz de gestionar los patrones de proyecto durante las fases del ciclo de vida.

En la Figura 12 se presenta de forma esquemática los elementos que componen sdpFramework.

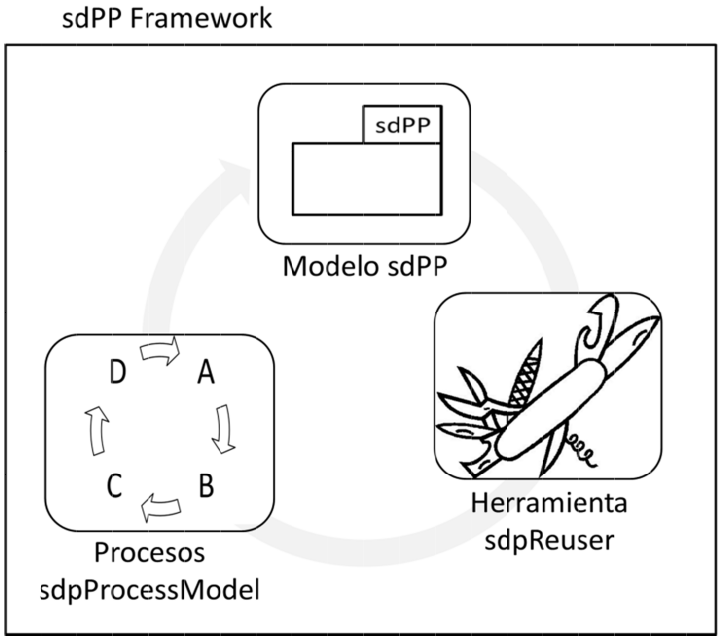


Figura 12 Elementos de sdpFramework

La primera parte de sdpFramework consistió en definir unos procesos para la gestión del conocimiento en forma de patrones de proyecto sobre metodologías de desarrollo, procedentes metodologías de desarrollo, mejores prácticas, lecciones aprendidas, marcos de referencia y conocimiento tácito de organizaciones de desarrollo software. Con estos procesos se ofrece una cobertura integral para la gestión del conocimiento, desde el momento que se crea, hasta que se aplica, se reutiliza o se actualiza. El modelo de ciclo de vida propuesto para la gestión del conocimiento de gestión de patrones de proyectos de desarrollo software tiene ocho fases (Figura 14).

Para gestionar el flujo de conocimiento entre las distintas actividades de los procesos, se ha diseñado un modelo de conocimiento llamado sdPP, este modelo abstrae los elementos de conocimiento más importantes del área de gestión de proyectos de desarrollo software, para poder formalizar conocimiento de las metodologías de desarrollo, marcos de referencia, buenas prácticas, lecciones aprendidas y conocimiento tácito de la organización e ingenieros de software. Además este modelo está diseñado para ser eficiente, efectivo y que ayude al usuario en su uso.

Para poder gestionar los sdPPs durante el ciclo de vida, se ha diseñado y desarrollado una herramienta llamada sdpReuser que ofrece cobertura en las principales fases del ciclo de vida de los procesos de gestión propuestos. sdpReuser permite la creación, adaptación, utilización, búsqueda de sdPPs.

3.1.2 Alcance de sdpFramework

Se ha diseñado sdpFramework para poder gestionar el conocimiento de patrones de proyecto en todas las actividades del ciclo de vida propuesto, desde la adquisición de conocimiento hasta su uso, pasando por la búsqueda o la distribución del conocimiento. El tipo de conocimiento que se pretende dar cobertura con este marco metodológico son patrones de proyecto. Los patrones de proyecto pretenden modelar el conocimiento necesario para resolver un proyecto completo de desarrollo software desde las fases iniciales hasta la entrega del proyecto y el análisis post-mortem del mismo. El modelo de conocimiento que vamos a utilizar en concreto será sdPP (Software Development Project Pattern). sdPP es un modelo de conocimiento capaz de modelar los elementos de conocimiento más utilizados en el dominio del desarrollo de software. sdpReuser esta diseñado para soportar activos de procesos en los idiomas español e inglés, ya que los motores de indexación y búsqueda están diseñados para funcionar en estos dos idiomas.

3.2 El modelo de conocimiento sdPP

El modelo sdPP se define como un par problema-solución, como la mayoría de los patrones que existen en la literatura (Gamma, 1995). El parte del problema describe los tipos de los proyectos de desarrollo software para los que es recomendable la utilización del patrón; mientras que la solución describe las mejores prácticas para desarrollar el proyecto explicado en la parte de la descripción. La Figura 13 muestra el modelo de conocimiento de sdPP.

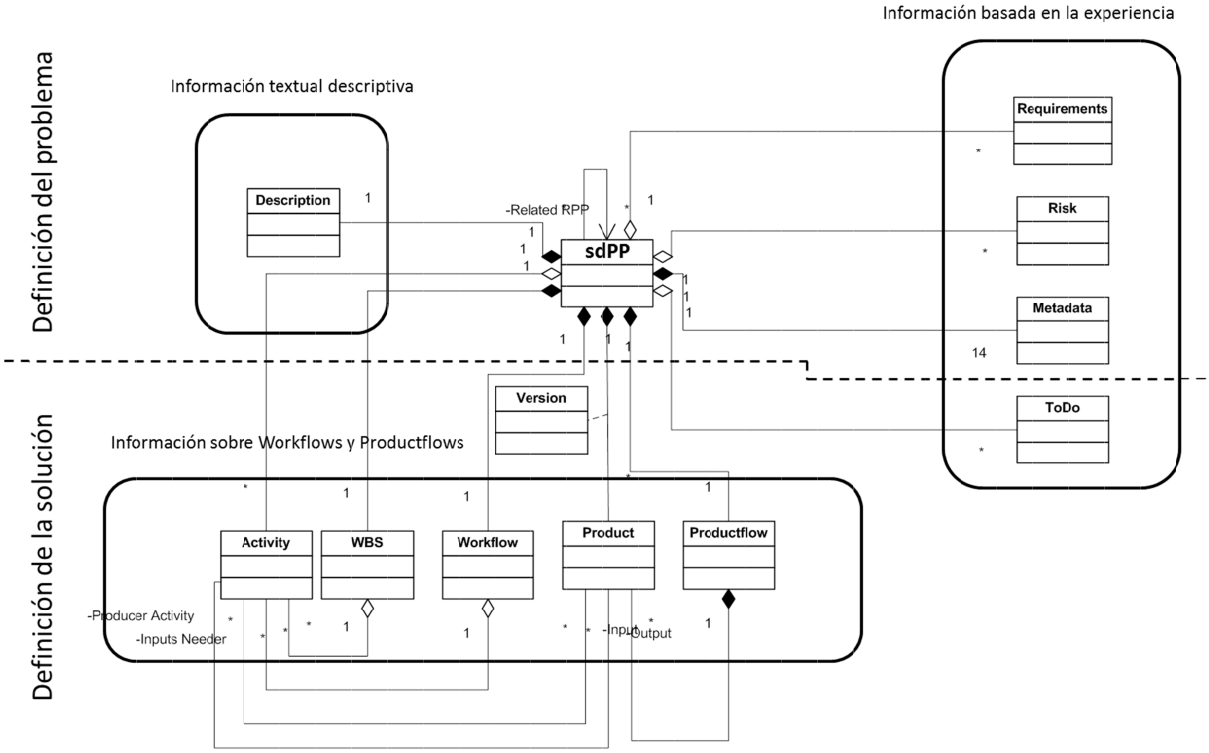


Figura 13 Modelo en UML sdPP

Para definir la parte del problema se proponen los siguientes elementos de información:

- Una descripción textual; que proporciona una explicación corta de los tipos de proyecto para los que es recomendable usar el sdPP.
- Un conjunto de metadatos; para etiquetar y clasificar el sdPP. Se han descrito catorce campos cuantitativos numéricos con valores entre 0 y 5. Estos catorce campos describen para que tipo de proyectos de desarrollo software es apropiado la aplicación del sdPP descrito.

En la parte de la solución se han descrito los siguientes elementos:

- Un WBS (Work Breakdown Structure); que define una estructura arbórea para organizar las actividades propuestas por el patrón. Se trata de una vista organizativa de las actividades que ayuda a los ingenieros de software a entender la jerarquía entre las actividades.
- Un workflow; que indica la secuencia recomendada para la realización las actividades descritas en el WBS. Se ha adoptado el diagrama de actividad de UML (OMG, 2007) para definir el workflow. El workflow guía a los ingenieros de software para

saber el orden adecuado o recomendado para ejecutar cada una de las actividades, además el workflow define la dependencia entre actividades.

- Un productflow, que describe como los productos fluyen entre las actividades. Típicamente, una actividad tiene un conjunto de productos de entrada (procedentes de la ejecución de una actividad) necesarios para su ejecución; y un conjunto de productos de salida que son el resultado de la ejecución de la actividad.
- Un conjunto de requisitos; que el gestor del proyecto debe satisfacer para poder aplicar la solución ofrecida por el patrón. Estos requisitos definen las precondiciones que se deben reunir para aplicar la solución que propone un sdPP.
- Un conjunto de riesgos; que el gestor de proyectos debe asumir si se aplica la solución propuesta por el sdPP. Estos riesgos alertan la los ingenieros de software de posibles problemas en la ejecución de un desarrollo software usando el sdPP concreto.
- Un conjunto de "to-dos"; con recomendaciones basadas en las mejoras prácticas y lecciones aprendidas. Esta lista de "to-dos" intenta formalizar el conocimiento tácito basado en la experiencia en metodologías de desarrollo, marcos de referencia y mejores prácticas. Un "to-do" no es una actividad o una parte del workflow (e.g. "trabajar en parejas" no es una actividad pero sin embargo es un "to-do").

En la Tabla 6 podemos ver la correspondencia que existe entre los elementos de información del modelo sdPP con el modelo canónico propuesto por Buschmann (Buschmann et al., 2007), cuyos elementos están descritos en la Tabla 2.

Elemento propuesto por el modelo canónico	**Correspondencia con el modelo sdPP**
Nombre	Nombre
Problema	Descripción
Contexto	Metadatos
Restricciones	Requisitos, Riesgos
Solución	Actividades, WBS, Workflow, Productflow
Ejemplos	Proyectos realizados con el sdPP concreto
Contexto resultante	Proyectos desarrollados
Razón	Descripción
Patrones relacionados	N/A
Usos conocidos	Proyectos realizados

Tabla 6 Correspondencia de los elementos de información del modelo canónico y el modelo sdPP

En los anexos se presenta un ejemplo de un sdPP, además existen más sdPPs de ejemplo en la siguiente dirección web:

http://www.diego-martin.info/sdPP-example

3.3 Procesos de sdpFramework: sdpProcessModel

La implantación de sdPP en una organización debe ser un proceso gradual que consiste varias actividades basadas en la integración e interacción de características de sistemas de gestión del conocimiento. Se ha desarrollado una herramienta llamada sdpReuser para gestionar y aplicar los sdPPs.

3.3.1 Descripción de los procesos de sdpFramework.

En este apartado se van presentar el conocimiento sobre los procesos del framework ofrecido en esta tesis. Para representar estos procesos vamos a utilizar el meta-modelo SPEM (OMG, 2008). SPEM (Software Process Engineering Meta-Model, Meta-Modelo para la Ingeniería del Proceso Software) es una especificación del grupo de gestión de objetos OMG (Object Management Group) utilizada para definir procesos de desarrollo de sistemas y software, así como sus componentes. Su objetivo principal es abarcar un amplio rango de métodos de desarrollo y procesos de diferentes estilos, niveles de formalismo y modelos de ciclo de vida, entre otros.

En la descripción del framework descrito en ésta tesis doctoral se utiliza SPEM 2.0 que es un tipo de modelo usado en la ingeniería del software y sistemas de ingeniería para el análisis y la construcción de modelos y útil para problemas predefinidos. Para la descripción de la arquitectura de sdpFramework se utilizarán los iconos propuestos por SPEM y descritos en la Tabla 7.

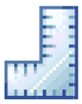
Nombre	Icono	Descripción
Proceso		Un proceso es un componente desglosable y ejecutor de la definición del trabajo que representa una relación entre las instancias de actividades y el uso de los roles de las instancias.
Actividad		Una actividad es una definición concreta de trabajo que representa una unidad general de trabajo asignable a ejecutores específicos por el uso de uno o más roles.
Tarea		Una tarea es un elemento que contiene el método y la definición del trabajo, y que indica como el trabajo es ejecutado por los roles. Una tarea está asociada a productos de trabajo de entrada y salida.
Producto		Un producto de trabajo es un elemento que es usado, modificado y generado por las tareas.
Rol		Un rol es un elemento que se define como un conjunto de habilidades, competencias y responsabilidades. Los roles son usados por las tareas para definir quién las ejecuta, así como también para definir un conjunto de productos de trabajo de los cuales está encargado.
Guía		Una guía es un elemento describible que proporciona información adicional relacionada con actividades y tareas. Algunos ejemplos de guías pueden ser directrices, plantillas, técnicas, mecanismos, etc.
Herramienta		Una herramienta es un elemento de contenido de método que puede ser usado para especificar la participación y capacidades de herramientas y en la definición de una tarea.

Tabla 7 Iconos de los estereotipos definidos por SPEM.

El conjunto de procesos propuestos en la presente tesis está compuesto por seis fases: Adquisición, preservación, adaptación, adaptación, utilización, distribución y medición. En los siguiente puntos se analizan y describen cada una de éstos procesos, mostrando en SPEM los actividades, tareas, roles y productos.

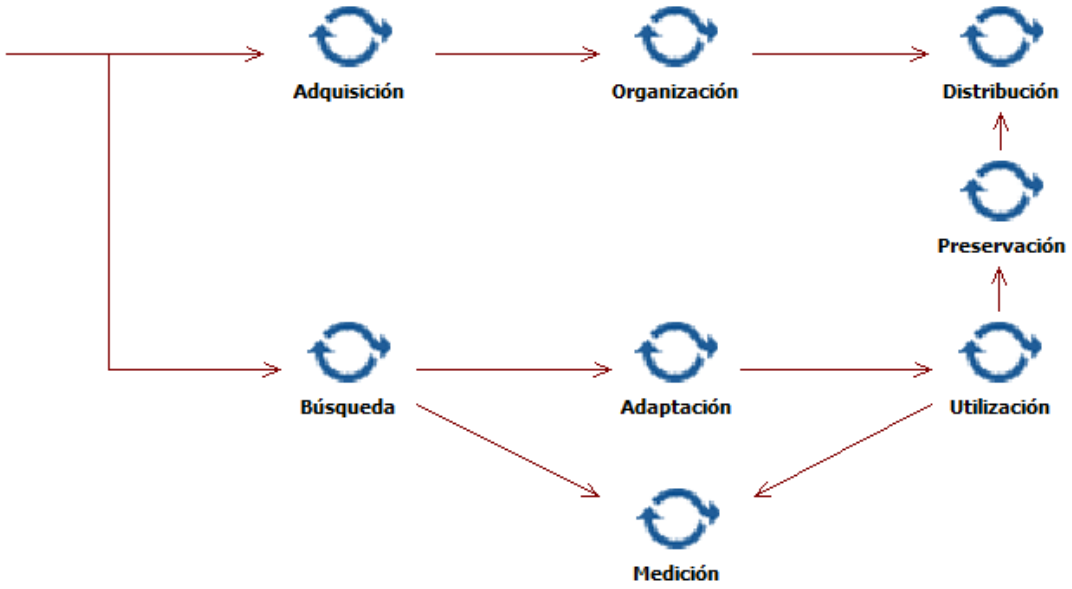


Figura 14 Modelo en SPEM de fases de los procesos de sdpFramework

En la Tabla 8 se mostrarán los roles existentes en los procesos de sdpFramework, además se mostrará de que actividades son responsables cada uno de los roles.

Rol	**Fases en las que es responsable**
Ingeniero de procesos	Adquisición, preservación, adaptación, utilización, búsqueda, medición, organización y distribución
Ingeniero de conocimiento	Preservación, adaptación, utilización, distribución
Ingeniero de software	Utilización, búsqueda

Tabla 8 Roles y responsabilidades

3.3.2 Adquisición de conocimiento

En este apartado se describirán las actividades que hay que llevar a cabo en la fase de adquisición de conocimiento, sus actividades y los productos de entrada y de salida.

3.3.2.1 Objetivo de la fase de adquisición de conocimiento.

Esta es la primera función disponible en sdpFramework y permite la implementación de la codificación de conocimiento y estrategias de personalización. Esta fase empieza con la necesidad de modelar la información para poder preservarla. Para empezar, el usuario debe buscar fuentes de conocimiento, ya sean tácitas o explicitas. El siguiente paso es seleccionar un conjunto de fuentes bibliográficas para modelar el conocimiento. A continuación el usuario debe entender la información. Cuando el usuario tiene suficientes fuentes bibliográficas y ha comprendido la información, puede empezar el modelado. sdpReuser permite la formalización de información ya que gestiona sdPPs, permitiendo su creación, edición y borrado. Podemos ver una descripción de la fase de adquisición de conocimiento en la Figura 15

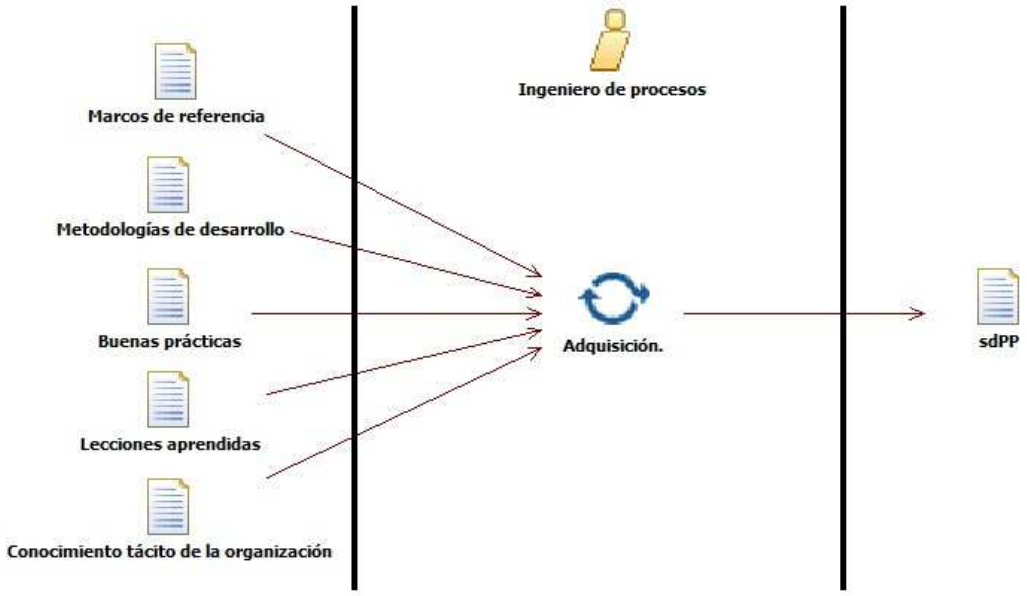


Figura 15 Fase de Adquisición, entradas salidas y roles

3.3.2.2 Entradas

Los productos de entrada optativos para poder ejecutar la fase actual son:

- Marcos de referencia
- Metodologías de desarrollo
- Buenas prácticas
- Lecciones aprendidas
- Conocimiento tácito de la organización

3.3.2.3 Roles

El rol responsable de esta actividad es: Ingeniero de procesos

3.3.2.4 Actividades a realizar

Las actividades que se deben realizar en la fase de adquisición de conocimiento son:

- Identificación de las referencias.
- Modelado de la parte del problema del patrón.
- Modelado de la parte de la solución del patrón.

Podemos ver una secuencia de las actividades asociadas a la fase de adquisición de información en la Figura 16

Figura 16 Fase de Adquisición: Actividades

A continuación se mostrará un diagrama de entrada/salida por cada una de las actividades de esta fase.

Identificación de las referencias

En esta actividad los ingenieros de procesos tienen que localizar las referencias más relevantes para poder extraer el conocimiento que será modelado en un sdPP. Las referencias pueden proceder de marcos de

referencia, metodologías de desarrollo, buenas prácticas, lecciones aprendidas o conocimiento tácito de la organización. En la Figura 17 podemos ver un diagrama de entradas/salidas de la actividad.

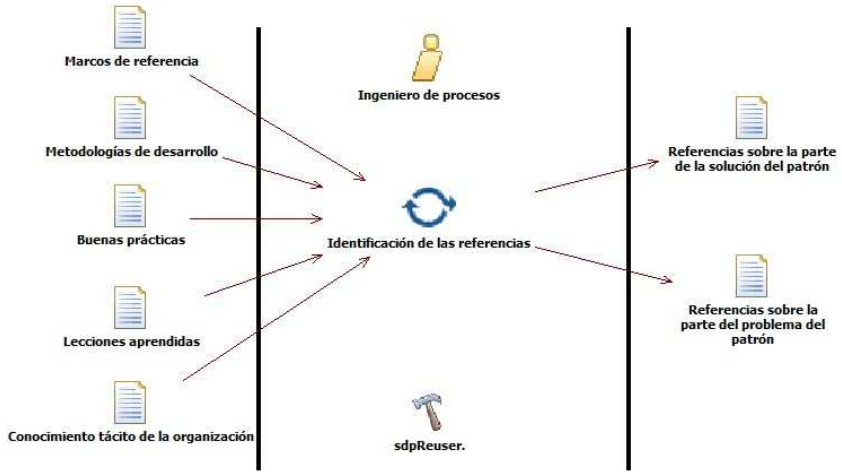


Figura 17 Actividad "Identificación de las referencias", entradas, salidas y roles

Modelado de la parte del problema del patrón

En esta actividad los ingenieros de procesos deben modelar la parte del patrón que se refiere a la descripción del problema. Los elementos de información que se corresponden con la descripción del problema son: "Descripción", "Metadatos" y "Requisitos". Se puede ver una descripción de esta actividad en un diagrama de entradas/salidas de SPEM en la Figura 18.

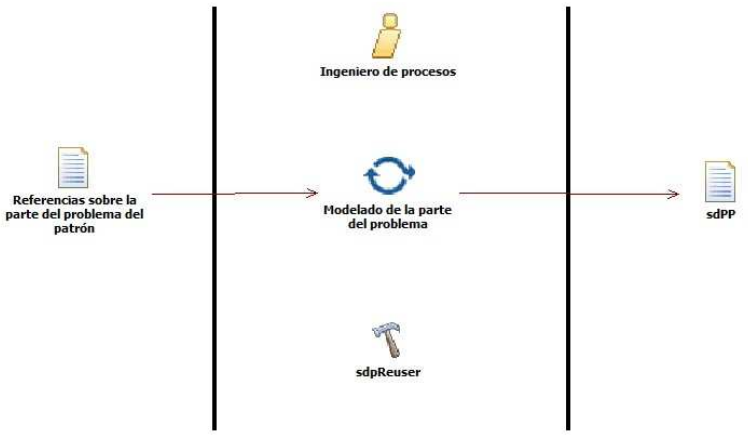


Figura 18 Actividad "Modelado de la parte del problema del patrón", entradas, salidas y roles

Modelado de la parte de la solución del patrón

En esta actividad los ingenieros de procesos deben modelar la parte del patrón que se refiere a la descripción de la solución. Los elementos de información que se corresponden con la descripción del problema son: "Workflow", "Productflow", "Riesgos", "To-dos". Se puede ver una descripción de esta actividad en un diagrama de entradas/salidas de SPEM en la Figura 19.

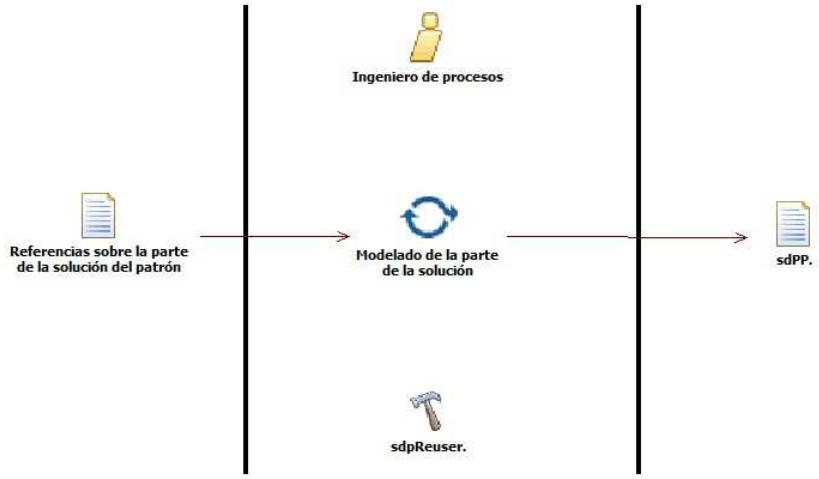


Figura 19 Actividad "Modelado de la parte de la solución del patrón", entradas, salidas y roles

3.3.2.5 Criterios de validación.

Para asegurar que la fase de Adquisición se realiza completamente se debe asegurar las siguientes preguntas:

- ¿Se ha realizado una buena selección de las fuentes bibliográficas?
- ¿Las referencias cubren las necesidades de información de la parte de la descripción del problema del patrón?
- ¿Las referencias cubren las necesidades de información de la parte de la descripción de la solución del patrón?

3.3.2.6 Salidas

El producto final de esta fase es un sdPP completo y correcto, esto quiere decir que todos los elementos de información que componen el patrón sdPP deben estar completados y con información correcta.

3.3.3 Búsqueda de conocimiento

3.3.3.1 Objetivo de la fase de Búsqueda

La herramienta sdpReuser proporciona facilidades para realizar búsquedas efectivas y navegación entre los sdPPs disponibles al personal que trabaja para una organización de desarrollo software. sdpReuser gestiona una estructura de información con múltiples tipos de nodos interconectados entre ellos que permiten búsquedas y navegabilidad entre ellos. Muchos elementos de sdPP se centran en conocimiento procedimental que implican: que acciones hay que tomar, quién las realiza, cuándo se han realizado y cómo han sido implementadas.

sdpReuser proporciona mecanismos para la búsqueda, recuperación y gestión de los activos correspondiente a procesos de desarrollo software. En la Figura 20 podemos ver de forma gráfica un resumen de los tipos de búsquedas de sdPPs.

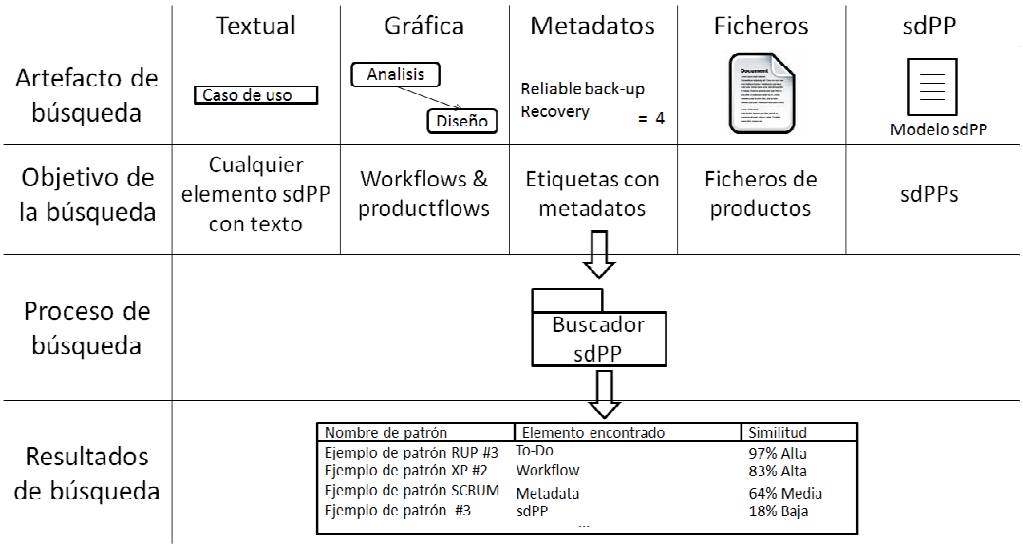


Figura 20 Búsquedas de sdPPs

El primero consiste en la recuperación textual a través de cualquier elemento de sdPP que contenga algún texto como, nombre de actividades, descripción, productos, "to-dos" y riesgos. Por ejemplo, el usuario puede buscar por nombres de actividades, nombres de productos o entre el texto de la descripción; sdpReuser recuperará elementos que contengan ese texto exactamente o similares.

El usuario puede realizar búsquedas a través de diagramas tales como el workflow o productflow para poder buscar sdPPs con diagramas similares.

Para realizar una búsqueda gráfica el usuario puede crear un diagrama, el cuál es el elemento de búsqueda y sdpReuser buscará a través del conjunto de patrones que tiene en su base de conocimiento. sdpReuser no solo recupera los sdPPs que contengan exactamente los elementos de la búsqueda, si no que recuperará sdPPs que sean similares.

Otra forma de buscar es realizando una "búsqueda mediante ejemplo" usando los productos como ficheros, documentos, hojas de cálculo para buscar sdPPs con productos similares (e.g. buscar patrones que contengan un fichero concreto). sdpReuser indiza toda la información sobre los sdPPs incluidos los productos. En muchos casos los productos son plantillas cuando el sdPP es un patrón o productos con información cuando los patrones son instanciados como un proyecto concreto. Por lo tanto, es posible usar un fichero con un formato bien conocido como por ejemplo PDF, XSLX, DOCX para realizar búsquedas sobre sdPPs que contengan esos ficheros s o ficheros similares.

Podemos ver una descripción de la fase de distribución de conocimiento en la Figura 21.

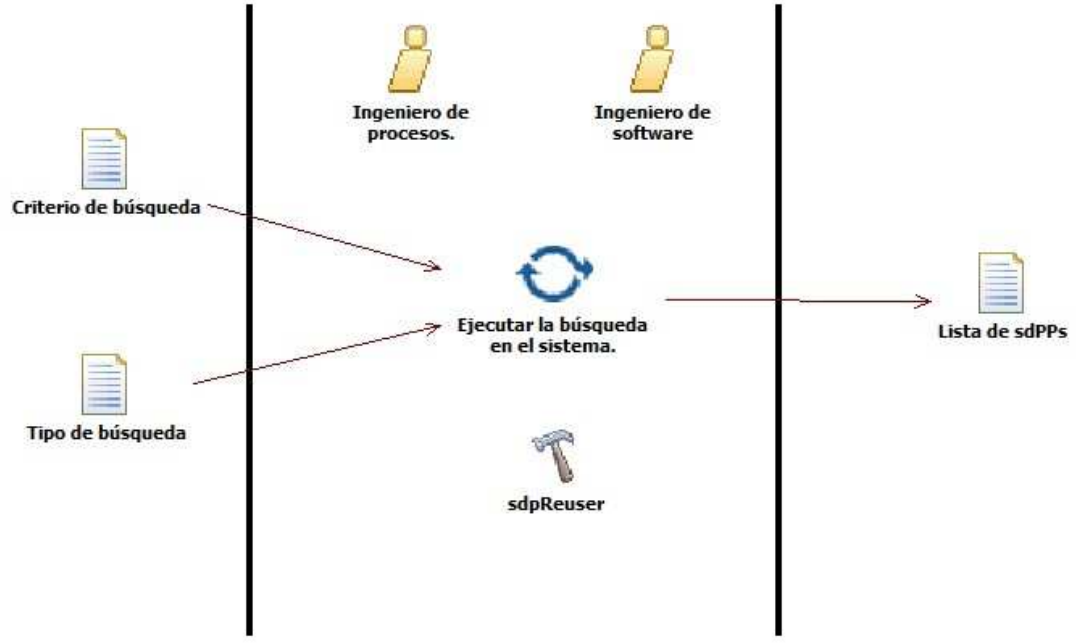


Figura 21 Fase de Búsqueda, entradas salidas y roles

3.3.3.2 Entradas

La única entrada de esta fase son los criterios de búsqueda sobre los que se quiere recuperar información.

3.3.3.3 Roles

El rol que interviene en esta fase el gestor de proyectos que debe elegir el sdPP que mejor se ajuste para poder ejecutar el proyecto de desarrollo.

3.3.3.4 Actividades a realizar

En esta actividad solo se ejecuta una acción que es la realización de la búsqueda en el sistema.

3.3.3.5 Criterio de validación

La fase de búsqueda se habrá realizado de forma correcta si se cumplen que el sdPP seleccionado es el que mejor se ajusta a los criterios de búsqueda.

3.3.3.6 Salidas

La salida de esta fase es una lista de sdPPs que se ajustan a los criterios de búsqueda.

3.3.4 Preservación de conocimiento

3.3.4.1 Objetivo de la fase de preservación

Esta fase se centra en mantener un repositorio de sdPPs, que además se van mejorando mediante la retroalimentación y los comentarios de los usuarios durante su uso de los sdPPs y mediante su experiencia adquirida en otros proyectos. sdpReuser permite a los usuarios añadir o actualizar los elementos de conocimiento de sdPP.

sdpReuser permite la creación de un sdPP desde cero, crear uno nuevo desde otro sdPP o convertir un proyecto concreto en un patrón nuevo. Cuando un equipo de desarrollo ha completado un proyecto, pueden crear un nuevo sdPP desde el proyecto concreto si ellos consideran apropiado para futuros desarrollos. En este caso, sdpReuser considerará el proyecto actual con los cambios realizados como un nuevo sdPP.

sdpReuser asegura la integridad del conocimiento a través del tiempo. Los sdPP desactualizados, anticuados u obsoletos serán identificados por los informes de uso ofrecidos por sdpReuser; basándose en estos informes el equipo de procesos decide si eliminar un sdPP. La eliminación controlada de patrones favorece la mejora continua del proceso de desarrollo. Podemos observar una descripción de esta fase en la Figura 22

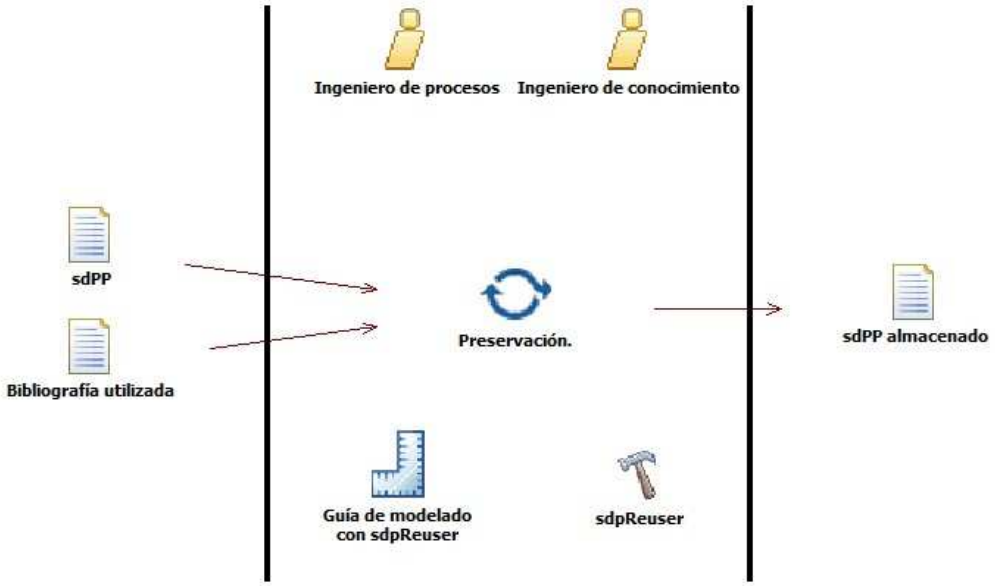


Figura 22 Fase de Preservación, entradas, salidas y roles

3.3.4.2 Entradas

Los siguientes productos son las entradas de la actividad:

- sdPP
- Bibliografía utilizada

3.3.4.3 Roles

En la siguiente fase nos encontraremos los siguientes roles:

- Ingeniero de procesos
- Ingeniero de conocimiento

3.3.4.4 Actividades a realizar

Las actividades a realizar en esta fase son:

- Comprobar que se sigue el formato convenido
- Almacenar de forma persistente en el sistema

En la Figura 23 se puede ver una secuencia de las actividades asociadas a la fase de preservación de la información.

Figura 23 Fase Preservación: Actividades

A continuación se mostrará un diagrama de entrada/salida por cada una de las actividades de esta fase.

Comprobar que el sdPP sigue el formato convenido

En esta actividad los ingenieros de conocimiento deben garantizar que el sdPP de entrada cumple el formato convenido descrito por el modelo de conocimiento sdPP, para ello los ingenieros de conocimiento podrán ponerse de acuerdo con los ingenieros de procesos para poder solucionar problemas de tipos de datos. La Figura 24 muestra un diagrama de entrada/salida describiendo la actividad.

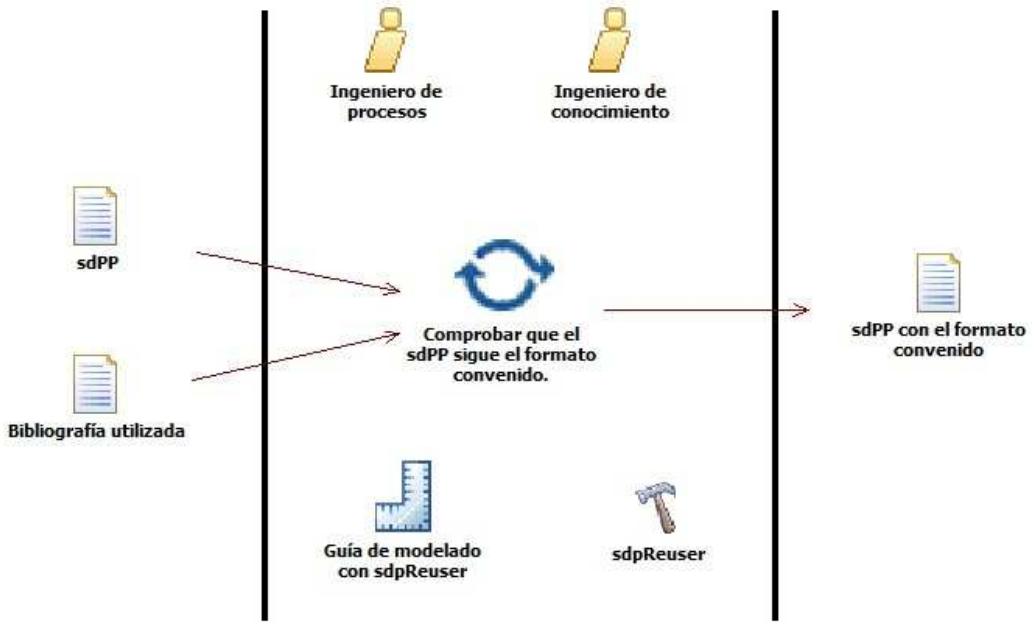


Figura 24 Actividad "Comprobar que el sdPP sigue el formato convenido"

Almacenar de forma persistente

Esta actividad consiste en almacenar de forma persistente la información de un sdPP concreto para garantizar su distribución a través de búsquedas y su posterior uso. La Figura 25 muestra un diagrama de entrada/salida en SPEM.

Figura 25 Actividad "Almacenar de forma persistente", entradas, salidas y roles

3.3.4.5 Criterios de validación

Para asegurar que esta fase se complete correctamente hay que asegurar las siguientes preguntas:

- ¿Se ha conseguido realizar una descripción precisa del sdPP en lenguaje natural con un tamaño aproximado de 250 caracteres?
- ¿Se ha clasificado las actividades en un WBS?
- ¿Se ha modelado el orden de las actividades en un workflow correctamente?
- ¿Se ha modelado los productos de cada una de las actividades de forma correcta?
- ¿Se han identificado correctamente los "to-dos" sin confundirlos con actividades?
- ¿Se ha clasificado el sdPP correctamente con metadatos?

- ¿Se han identificado correctamente los requisitos necesarios para poder acometer el proyecto que representa el sdPP?
- ¿Se han identificado correctamente los riesgos que hay que asumir a la hora de elegir el sdPP para acometer un proyecto?

3.3.4.6 Salidas

La salida de esta actividad consiste en un sdPP modelado en el sistema informático y almacenado de forma persistente, para poder en futuras actividades distribuirlo, realizar búsquedas entre distintos sdPP, utilizarlo etc.

3.3.5 Adaptación de conocimiento

3.3.5.1 Objetivo de la fase de adaptación

Los usuarios pueden recuperar y reutilizar sdPPs para ser aplicados a un proyecto concreto. La información que contiene un patrón de proyecto sdPP debe ser acondicionada al entorno donde se va a desarrollar el proyecto, además de ajustarse a los requisitos concretos del desarrollo. De esta manera cualquier patrón puede ser adaptado para ajustarse a un contexto específico. sdpReuser ofrece la posibilidad de adaptar y ajustar cualquier sdPP a cualquier escenario de desarrollo de software. Podemos ver una descripción de la fase de adquisición de conocimiento en la Figura 26

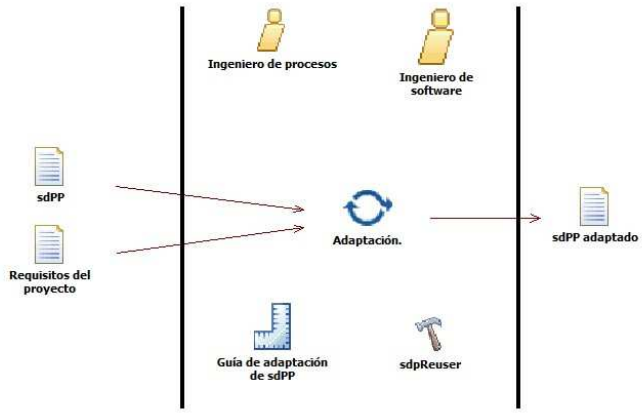


Figura 26 Fase de Adaptación, entradas salidas y roles

3.3.5.2 Entradas

Las entradas de esta actividad son:

- Un modelo sdPP original
- Requisitos del proyecto al que se quiere adaptar

3.3.5.3 Roles

Los roles necesarios para esta actividad son:

- Ingeniero de software
- Ingeniero de procesos

3.3.5.4 Actividades a realizar

Las actividades a realizar en esta actividad son:

- Análisis de requisitos del proyecto nuevo
- Análisis del sdPP original
- Adaptar cambios al sdPP

Podemos ver un gráfico de actividades de esta fase en la Figura 27

Figura 27 Fase Adaptación: Actividades

A continuación se mostrará un diagrama de entrada/salida por cada una de las actividades de esta fase.

Análisis de requisitos del proyecto nuevo

Para poder aplicar el conocimiento de un sdPP se debe poder garantizar que éste se puede adaptar a los nuevos requisitos contextuales del proyecto de desarrollo al que se quiere adaptar. En esta actividad el ingeniero de software analiza si es posible la adaptación a los nuevos requisitos contextuales del proyecto de desarrollo. La Figura 28 muestra una descripción de esta actividad con un gráfico de entrada/salida de SPEM.

Figura 28 Actividad "Análisis de requisitos del proyecto nuevo", entradas, salidas y roles

Análisis del sdPP original

Antes de poder adaptar el sdPP hay que comprender y entender el conocimiento que éste almacena, por lo tanto un ingeniero de procesos debe estudiar dicho conocimiento. La Figura 29 muestra un diagrama de entrada/salida de esta actividad.

Figura 29 Actividad "Análisis del sdPP original", entradas, salidas y roles

Adaptar cambios al sdPP

Una vez comprobado que el sdPP se puede adaptar a los nuevos requisitos contextuales de proyecto de desarrollo software y que se ha comprendido el conocimiento del sdPP se puede llevar a cabo la adaptación del conocimiento a los nuevos requisitos contextuales. En la Figura 30 podemos ver un diagrama entrada/salida que describe esta actividad.

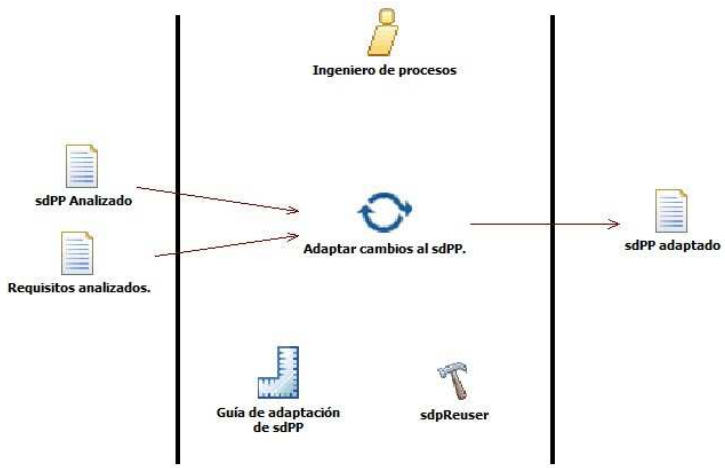


Figura 30 Actividad "Adaptar cambios al sdPP", entradas, salidas y roles

3.3.5.5 Criterios de validación

Los criterios de validación que debemos asegurar en esta fase son los siguientes:

- El sdPP creado debe ajustarse a las necesidades del nuevo proyecto de desarrollo software.
- El sdPP creado debe mantener un cierto parecido al sdPP original.

3.3.5.6 Salidas

La salida de esta actividad es un sdPP basado en el sdPP original pero adaptado a las nuevas necesidades del nuevo proyecto de desarrollo software.

3.3.6 Utilización de conocimiento

3.3.6.1 Objetivo de la fase de utilización

Cuando los usuarios han encontrado el sdPP apropiado y lo han adaptado a las necesidades del proyecto, pueden instanciarlo como un proyecto concreto y cambiar cada ítem de acuerdo al contexto específico del proyecto. sdpReuser guía al usuario a través de las actividades que el sdPP escogido recomienda e indica los productos necesarios para ejecutar una actividad; tanto los productos de entrada para ejecutar la actividad, como los productos de salida generados en la actividad.

sdpReuser ofrece herramientas para que los desarrolladores de software puedan realizar el proyecto de desarrollo software bajo las recomendaciones del sdPP. sdpReuser ayuda al guiado entre las actividades que propone el sdPP indicando cuales son las que se deben realizar en cada momento además de informar de los productos de entrada requeridos para poder acometer cada una de las actividades y los productos de salida que deben generar las actividades. Por cada una de estas actividades un sdPP ofrece una guía para poder realizar la actividad y unas plantillas de los productos que hay que realizar.

El sdPP ofrece unas plantillas básicas por cada producto que hay que completar en cada actividad.

La Figura 31 Fase de Utilización, entradas salidas y roles muestra un diagrama de entrada y salida de la fase de utilización.

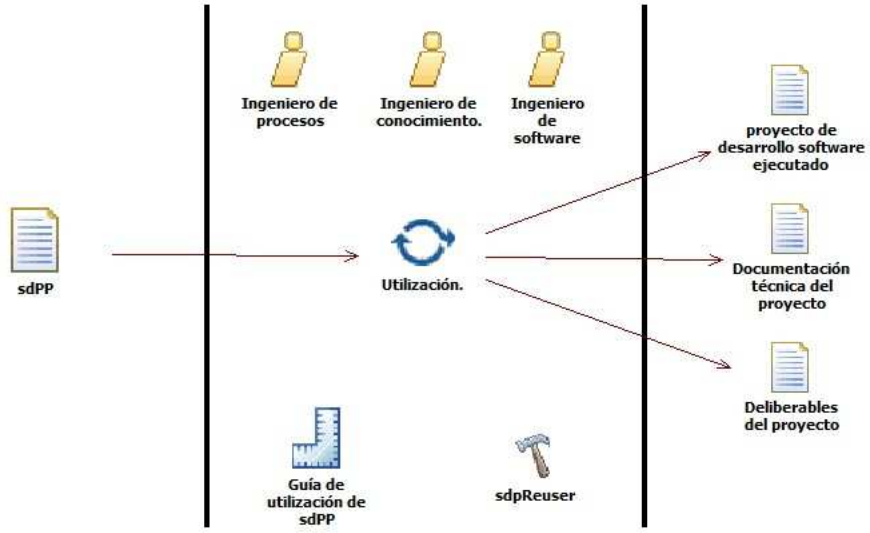


Figura 31 Fase de Utilización, entradas salidas y roles

3.3.6.2 Entradas

La entrada de esta fase es el sdPP seleccionado para ejecutar el proyecto de desarrollo software y que además ha sido adaptado para ajustarse al proyecto concreto y su entorno.

3.3.6.3 Roles

Los roles que van a intervenir en esta actividad son:

- Desarrollador de software
- Gestor de proyectos
- Ingeniero de proyectos

3.3.6.4 Actividades a realizar

Las actividades necesarias para completar esta actividad son:

- Asegurar que se cumplen los requisitos propuestos por el sdPP
- Controlar los riesgos propuestos por el sdPP
- Considerar los "to-dos"
- Ejecutar actividades según orden marcado por el workflow
- Finalizar proyecto

En la Figura 32 siguiente se muestra de forma gráfica la secuencia de actividades de la fase de utilización.

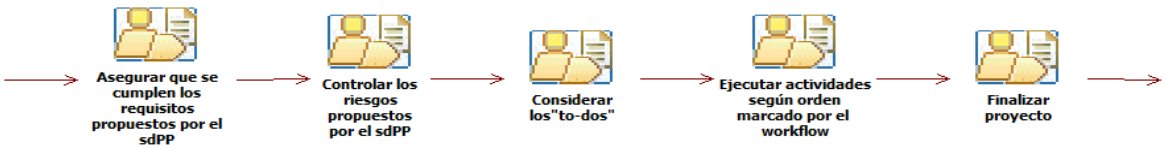


Figura 32 Fase Utilización: Actividades

A continuación se mostrará un diagrama de entrada/salida por cada una de las actividades de esta fase.

Asegurar que se cumplen los requisitos propuestos por el sdPP

En esta actividad el ingeniero de software debe asegurar que se cumplen los requisitos que impone el sdPP que se quiere utilizar. La Figura 33 muestra un diagrama entrada/salida en SPEM.

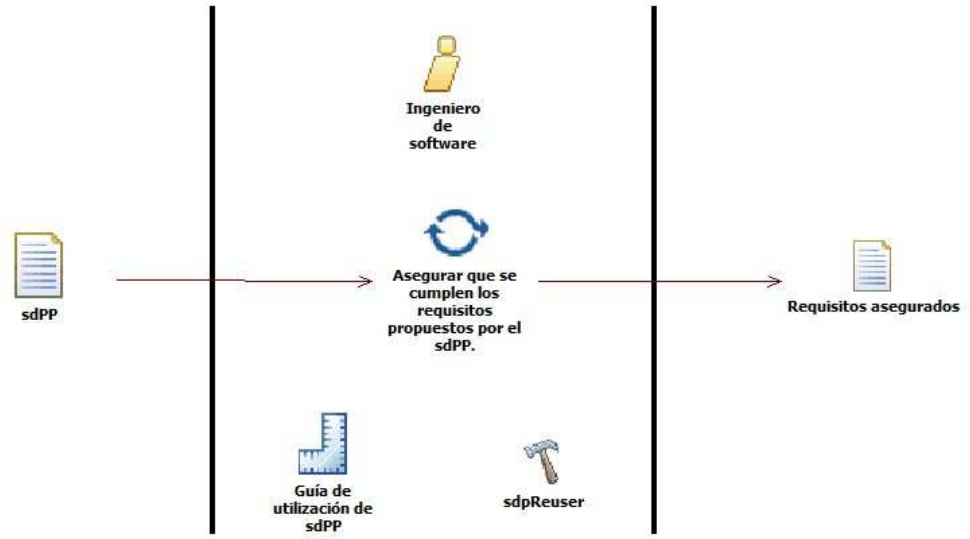


Figura 33 Actividad "Asegurar que se cumplen los requisitos propuestos por el sdPP", entradas, salidas y roles

Controlar los riesgos propuestos por el sdPP

En esta actividad el ingeniero de software debe controlar los riesgos que propone el sdPP. La Figura 34 muestra un diagrama entrada/salida de esta actividad.

Figura 34 Actividad "Controlar los riesgos propuestos por el sdPP", entradas, salidas y roles

Considerar los "to-dos"

El ingeniero de software debe considerar los "to-dos" que ofrece el sdPP, los "to-dos" deben cumplirse para asegurarse una correcta ejecución del proyecto de desarrollo software, pero no se tratan de actividades. En la Figura 35 podemos ver una descripción gráfica de la actividad con un diagrama de entrada/salida de SPEM.

Figura 35 Actividad "Considerar los to-dos", entradas, salidas y roles

Ejecutar actividades según orden marcado por el workflow

En esta actividad el ingeniero de software ejecuta las actividades propuestas por el sdPP, siguiendo el flujo que marca el workflow y creando los productos que indica el workflow. Podemos ver un diagrama de entrada/salida en la Figura 36.

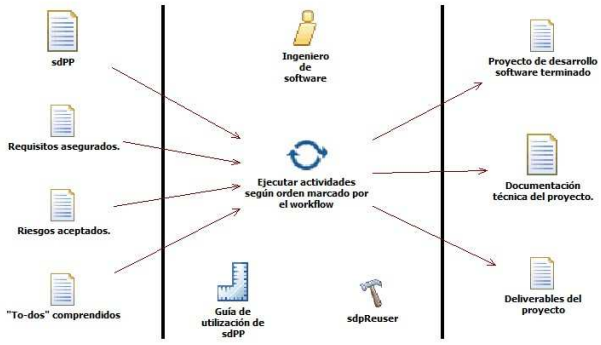


Figura 36 Actividad "Ejecutar actividades según orden marcado por el workflow", entradas, salidas y roles

Finalizar proyecto

El ingeniero de software debe marcar como finalizado el proyecto de desarrollo software, si el ingeniero de procesos así lo estima validará la finalización y el ingeniero de conocimiento estudiará si la ejecución del proyecto de desarrollo es relevante como para poder crear un nuevo sdPP a partir del proyecto de desarrollo ejecutado. En la Figura 37 podemos ver un diagrama de entrada/salida de esta actividad.

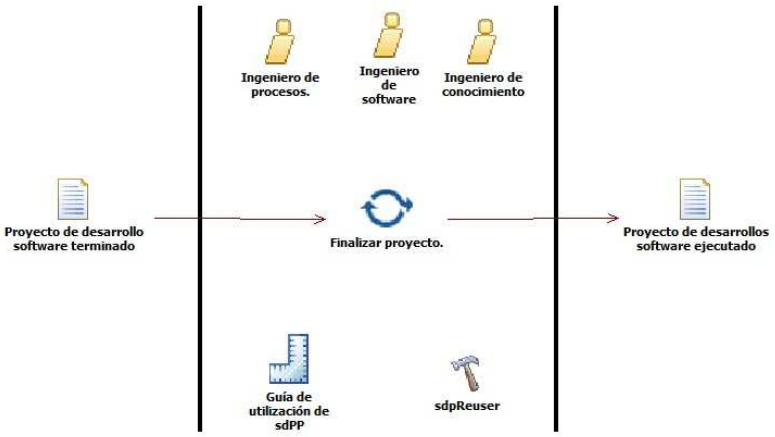


Figura 37 Actividad "Finalizar proyecto", entradas, salidas y roles

3.3.6.5 Criterios de validación

Para asegurarse la completitud de esta fase y que se ha realizado de forma correcta se deben responder de forma afirmativa a las siguientes preguntas de validación:

¿Se han ejecutado cada una de las actividades marcadas por el workflow del sdPP?

¿Se han realizado todos los productos marcados por el productflow?

¿Se ha ejecutado el proyecto de desarrollo siguiendo los consejos de los "to-dos"?

3.3.6.6 Salidas

Las salidas de la fase de utilización son:

- El proyecto de desarrollo software
- Documentación técnica del proyecto

- Entregables del proyecto

3.3.7 Medición de conocimiento

3.3.7.1 Objetivo de la fase de medición

Para evaluar de forma apropiada y efectiva la gestión del conocimiento de las organizaciones de gestión de procesos, sdpReuser proporciona funcionalidades para monitorizar el uso de los elementos de los sdPPs para poder identificar oportunidades para la mejora e introducir cambios correctivos. A través de la recopilación de medidas cuantitativas se pueden generar informes sobre los elementos de sdPP añadidos, adaptados y modificados durante la ejecución del proyecto de desarrollo.

Podemos ver una descripción de la fase de distribución de conocimiento en la Figura 38.

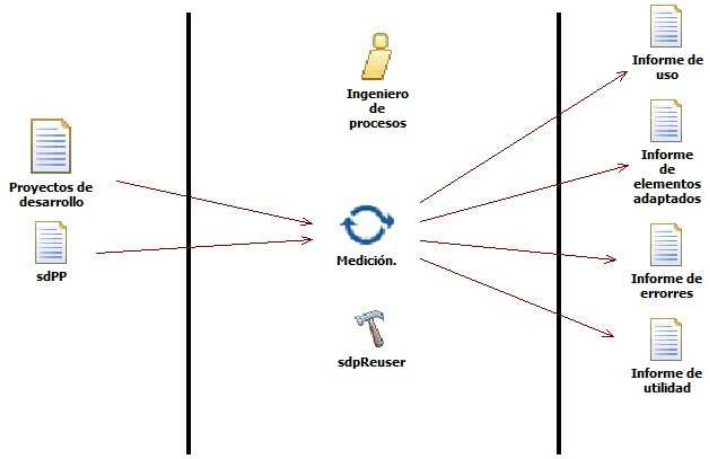


Figura 38 Fase de Medición, entradas salidas y roles

3.3.7.2 Entradas

Las entradas necesarias en esta fase son:

- El proyecto de desarrollo sobre el que se quiere hacer la medición
- El sdPP sobre el que se quiere hacer la medición.

3.3.7.3 Roles

El rol que interviene en esta fase es el gestor del proceso, que se encargará de analizar los informes generados en esta fase para poder realizar acciones correctivas.

3.3.7.4 Actividades a realizar

Las actividades a realizar en la fase de medición son:

- Seleccionar el indicador que se quiera medir.
- Ejecutar el algoritmo de medición.

En la Figura 39 podemos ver la secuencia de actividades necesarias para llevar a cabo esta fase.

Figura 39 Fase Medición: Actividades

A continuación se mostrará un diagrama de entrada/salida por cada una de las actividades de esta fase.

Seleccionar el indicador

El ingeniero de procesos debe seleccionar el indicador sobre el cual quiere realizar un estudio para crear un informe. La Figura 40 muestra un diagrama de entrada/salida de esta actividad.

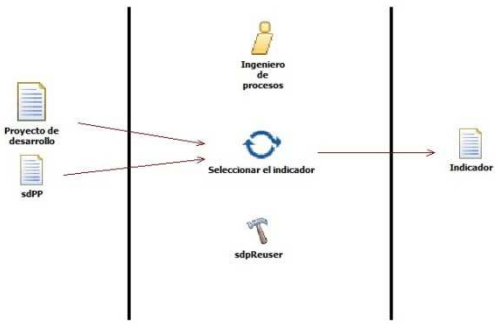


Figura 40 Actividad "Seleccionar el indicador que se quiera medir", entradas, salidas y roles

Ejecutar el algoritmo de medición

Cuando se ha seleccionado el criterio de medición se puede ejecutar el algoritmo que realiza los cálculos pertinentes para mostrar información relevante procesada de los datos recopilados en la ejecución de un proyecto de desarrollo software. La Figura 41 muestra un diagrama de entrada/salida.

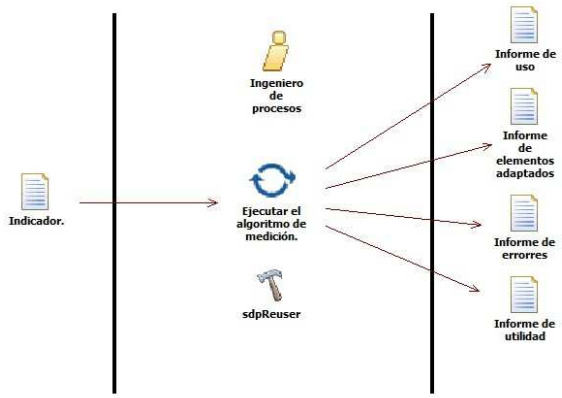


Figura 41 Actividad "Ejecutar el algoritmo de medición", entradas, salidas y roles

3.3.7.5 Criterios de validación

Esta fase estará completamente acabada si el ingeniero del proceso es capaz de analizar los indicadores seleccionados para poder realizar medidas correctivas a los sdPP en base a los informes generados.

3.3.7.6 Salidas

Las salidas de esta fase son:

- Informe de uso
- Informe de utilidad
- Informe de elementos adaptados
- Informe de errores

3.3.8 Organización de conocimiento

3.3.8.1 Objetivo de la fase de organización.

El modelo sdPP proporciona un conjunto de catorce metadatos que ayudan a la clasificación del patrón; sdpReuser permite la clasificación de

patrones dentro de una jerarquía de metodologías de desarrollo. Esa clasificación ayuda a ejecutar de forma efectiva búsquedas en la base de conocimiento. En la Figura 42 podemos ver una descripción de la fase de organización de conocimiento.

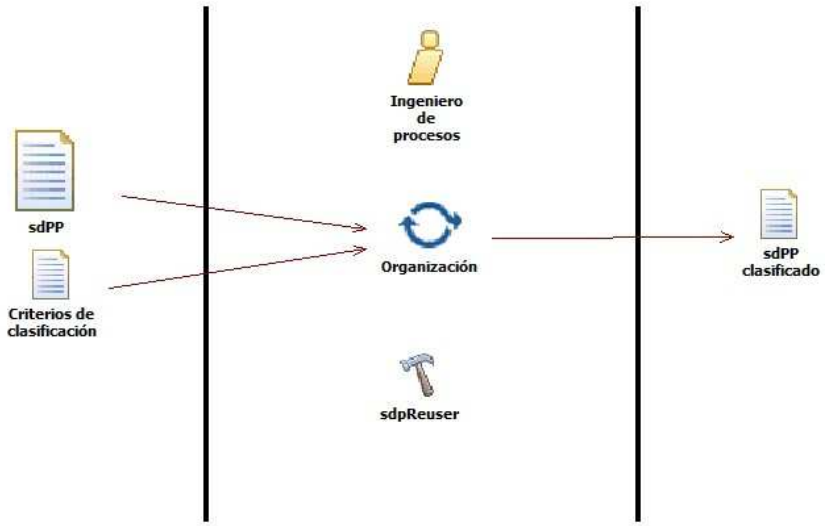


Figura 42 Fase de Organización, entradas, salidas y roles

3.3.8.2 Entradas

Las entradas de la fase de organización son:

- Un sdPP
- Los criterios de clasificación y organización

3.3.8.3 Roles

El rol responsable de esta actividad es el ingeniero de procesos.

3.3.8.4 Actividades a realizar

En la fase de organización es necesario realizar las siguientes actividades:

- Analizar los criterios de organización
- Clasificar el sdPP

En la Figura 43 se puede ver la secuencia de actividades necesarias para llevar a cabo esta fase.

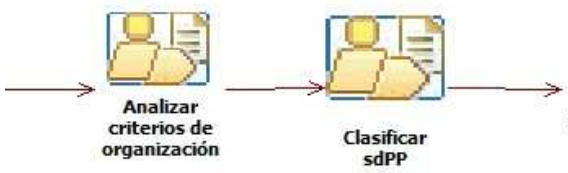


Figura 43 Fase de Organización: Actividades

A continuación se mostrará un diagrama de entrada/salida por cada una de las actividades de esta fase.

Analizar los criterios de organización

Para poder organizar correctamente la información el ingeniero de procesos debe decir cual es el criterio de organización por el cual se quiere clasificar el conocimiento de un sdPP concreto. La Figura 44 muestra un diagrama que explica esta actividad.

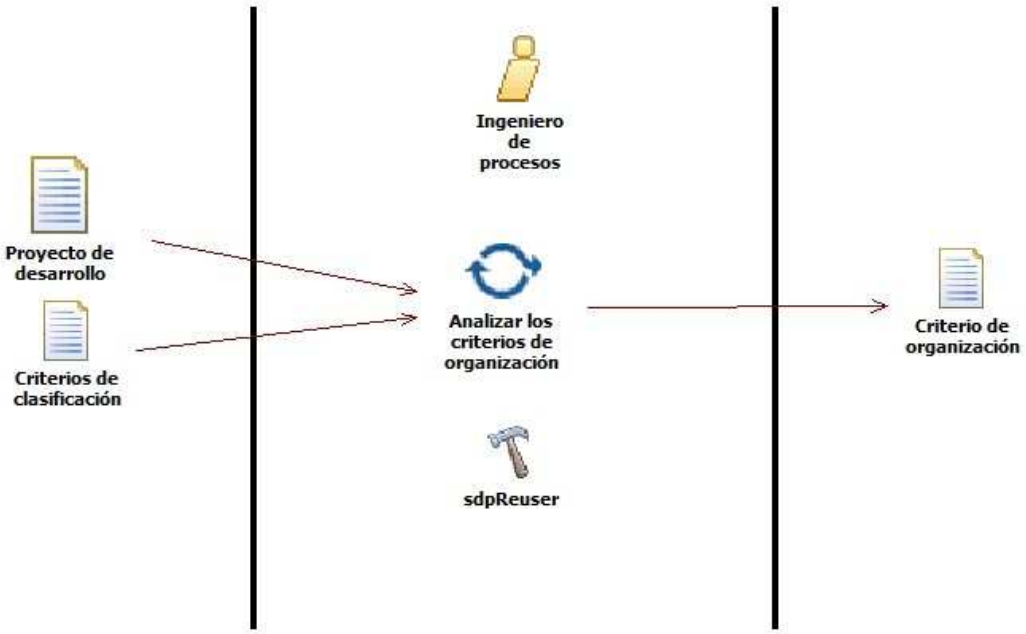


Figura 44 Actividad "Analizar los criterios de organización", entradas, salidas y roles

Clasificar el sdPP

Después de que el ingeniero de procesos ha elegido el criterio de clasificación, posteriormente deberá elegir como clasificar el sdPP bajo dicho criterio. La Figura 45 muestra un diagrama de entrada/salida de esta actividad.

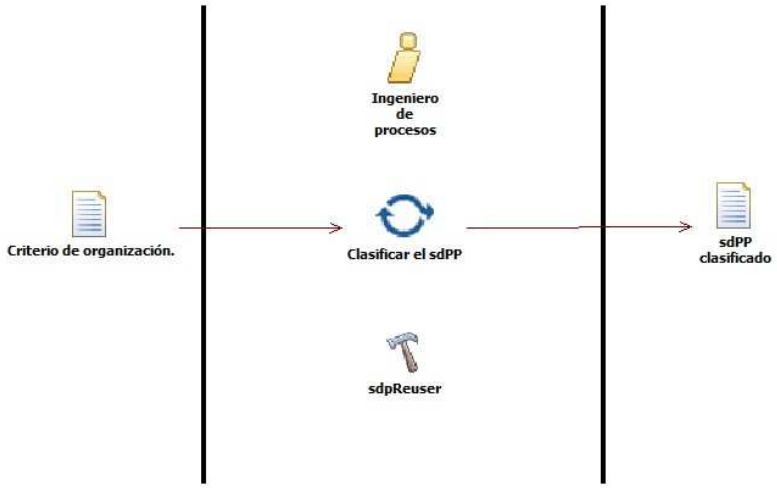


Figura 45 Actividad "Clasificar el sdPP", entradas, salidas y roles

3.3.8.5 Criterios de validación

Esta fase estará completamente finalizada cuando el ingeniero de procesos clasifique y organice el sdPP bajo los criterios de organización indicados.

3.3.8.6 Salidas

La salida de esta fase es un sdPP clasificado.

3.3.9 Distribución de conocimiento

3.3.9.1 Objetivo de la fase de distribución

El objetivo de la fase de distribución consiste en ofrecer el conocimiento de forma abierta y compartida a las personas o grupos de personas que lo requieran y lo necesiten. El método de compartir la información puede ser a través de una intranet para que solo los trabajadores de una organización puedan acceder a dicho conocimiento o a través de Internet para que pueda ser accedido por cualquier persona o buscador de información.

En la Figura 46 muestra una descripción gráfica de la fase de distribución de conocimiento.

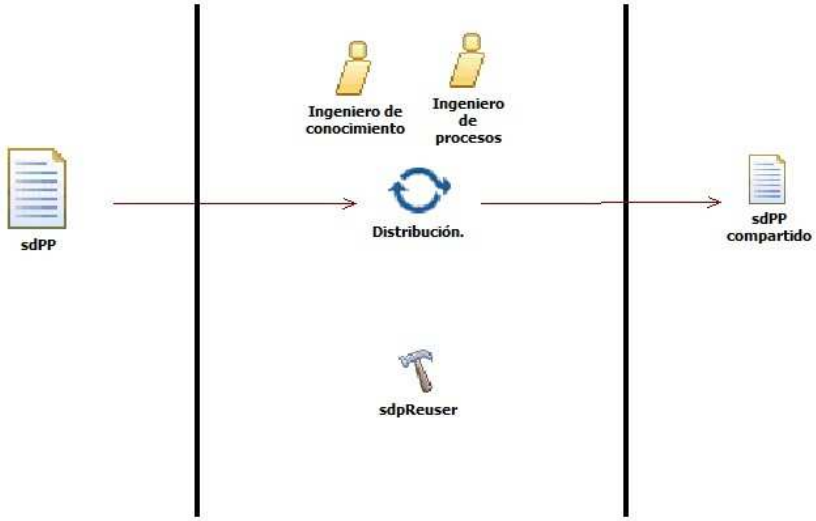


Figura 46 Fase de Distribución, entradas, salidas y roles

3.3.9.2 Entradas

La entrada de esta fase es el sdPP que se quiere distribuir o compartir.

3.3.9.3 Roles

Los roles implicados en esta fase son: Ingeniero de procesos e Ingeniero de conocimiento.

3.3.9.4 Actividades a realizar

La única actividad de esta fase consiste en compartir la información.

3.3.9.5 Criterios de validación

Para validar si esta fase ha sido correctamente realizada será necesario comprobar si el conocimiento del sdPP compartido esta accesible a las personas interesadas.

3.3.9.6 Salidas

La única salida de esta fase es un sdPP compartido y accesible por las personas interesadas.

3.4 Plataforma tecnológica sdpReuser

La Figura 47 muestra las funcionalidades de la herramienta sdpReuser.

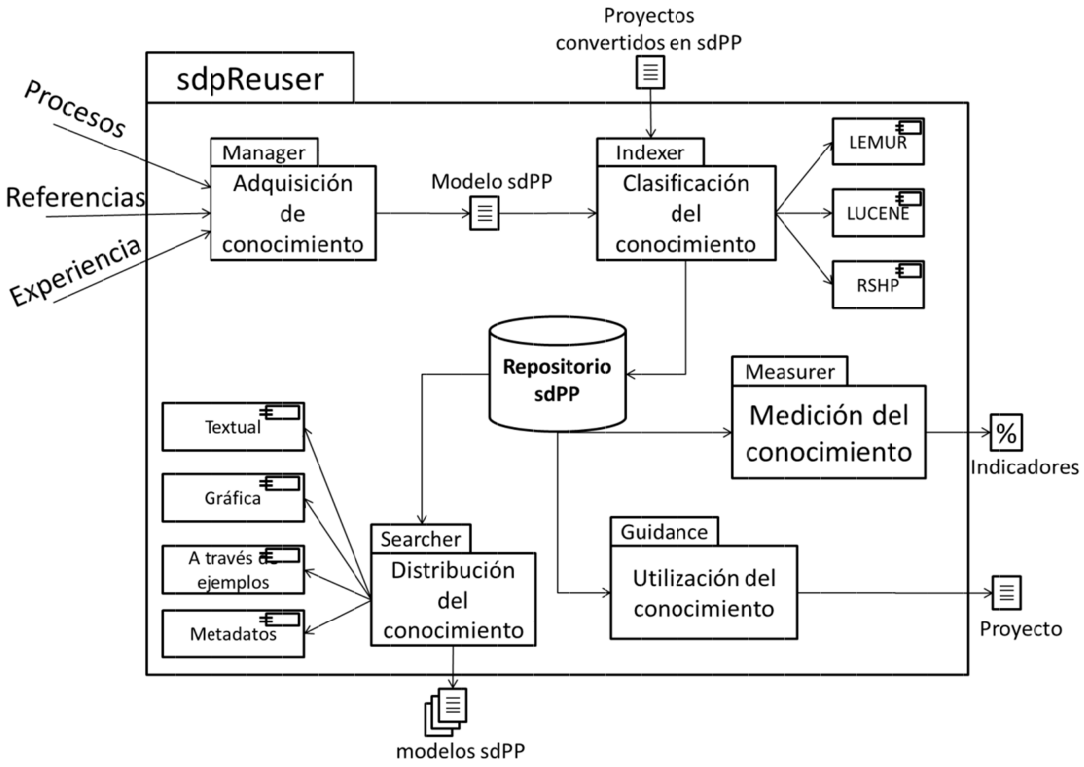


Figura 47 Funcionalidad de sdpReuser

Como podemos ver en la Figura 47, sdpReuser está dividido internamente en cinco paquetes, cada uno responsable de una funcionalidad casi siempre orientada a una fase del ciclo de vida.

1. Manager: Este módulo se encarga de la gestión de sdPP permitiendo la creación, modificación, adaptación y borrado de los mismos. La creación de los patrones se puede realizar a través de procesos desplegados en una empresa, conocimiento tácito de los ingenieros de software y referencias, tales como metodologías, buenas prácticas o lecciones aprendidas. También se puede crear un sdPP a partir de un proyecto concreto y convirtiendo en un patrón para futuros desarrollos. Este modulo se corresponde con la fase de adquisición y adaptación del ciclo de vida. La funcionalidad que ofrece este módulo es:
    - Creación de un sdPP
    - Modificado de un sdPP

- Borrado de un sdPP obsoleto
- Creación de un sdPP a partir de un proyecto de desarrollo software ejemplar que se quiera utilizar en un futuro como un sdPP
- Organización de los sdPP bajo un criterio organizativo
- Distribución de los sdPPs a través de una intranet o a través de internet, para garantizar que el conocimiento llega allá donde se necesite.

2. Indexer: Este módulo es capaz de indizar los elementos de un sdPP para poder realizar búsquedas precisas. Este módulo es capaz de indizar todos los elementos que utilicen lenguaje natural así como elementos gráficos o estructurados. En este módulo se utilizan los componentes Lucene (Apache Jakarta, 2005), Lemur (Allan, 2003) y RSHP (Llorens et al., 2004); que ofrecen la funcionalidad de poder indexar información en lenguaje natural y elementos estructurados. Este modulo ayuda en las siguientes fases del ciclo de vida: creación, adaptación, utilización y adaptación. La funcionalidad que ofrece este componente es:
   - Organizar el conocimiento de un sdPP en un repositorio
   - Garantizar que se puedan realizar todo de búsquedas sobre los sdPPs
3. Searcher: Con este módulo se pueden realizar búsquedas eficientes sobre el conjunto de patrones indexados y almacenados en el repositorio. Las búsquedas se pueden hacer por cualquiera los elementos de conocimiento que forman el sdPP, ya sean elementos que contengan información en lenguaje natural o elementos con información gráfica o estructurada. Además se puede realizar búsquedas ofreciendo ejemplos, y buscar por archivos de productos similares, así como realizar búsquedas por sdPPs parecidos a uno dado. Otra vía para poder realizar búsquedas eficientes es a través de los metadatos. Este módulo ayuda a las fases de distribución, búsqueda y utilización. Las funcionalidades que ofrece este componente son:
   - Realizar búsquedas textuales sobre cualquier elemento de sdPP que contenga un campo de texto. Este tipo de búsqueda no solo es textual si no que además es semántica.
   - Realizar búsquedas cuantitativas y cualitativas por metadatos.

- Realizar búsquedas "a través de ejemplo". El sistema es capaz de indizar todo tipo de productos, pudiéndose realizar búsquedas de sdPPs a través de ficheros.
- Realizar búsquedas gráficas. Un sdPP contiene un workflow y un productflow, estos elementos representan una abstracción de elementos que se pueden diagramar. Se pueden realizar búsquedas de los workflows y los productflows a través de un diagrama de ejemplo que hace las veces de artefacto de búsqueda.

4. Guidance: Este módulo se encarga de facilitar a los ingenieros de software la gestión de un proyecto de desarrollo software guiándolos a través de las fases que indica el workflow dentro del sdPP. Por cada una de las fases se ofrece unas plantillas con los productos que se tienen que desarrollar tal y como se describe en el productflow, y además por cada fase se ofrece un conjunto de guías y asistentes que ayudan a los ingenieros de software a realizarlas. Este módulo ayuda en la fase de utilización. Las funcionalidad de este componente son:
   - Guiar a los ingenieros de software que quieran acometer un proyecto de desarrollo software bajo las recomendaciones de un sdPP concreto, siguiendo el flujo de actividades del workflow; e indicando que productos son las entradas y las salidas de cada una de estas actividades según indica el productflow.
5. Measurer: Con este módulo se puede realizar diversas estadísticas sobre indicadores de proyectos para poder llevar a cabo un análisis post-mortem de cada uno de los proyectos y poder realizar medidas correctivas en caso de que un sdPP pueda inducir a errores y problemas en el desarrollo. Las funcionalidades de este componente es:
   - Seleccionar un indicador que se quiera medir en el proceso de desarrollo software
   - Ejecutar un algoritmo para extraer información a través de todos los datos adquiridos durante la ejecución del proyecto software.
   - Presentar informes con los indicadores seleccionados.

En la Figura 48 podemos ver un diagrama donde se observa qué componente de sdpReuser realiza qué actividad.

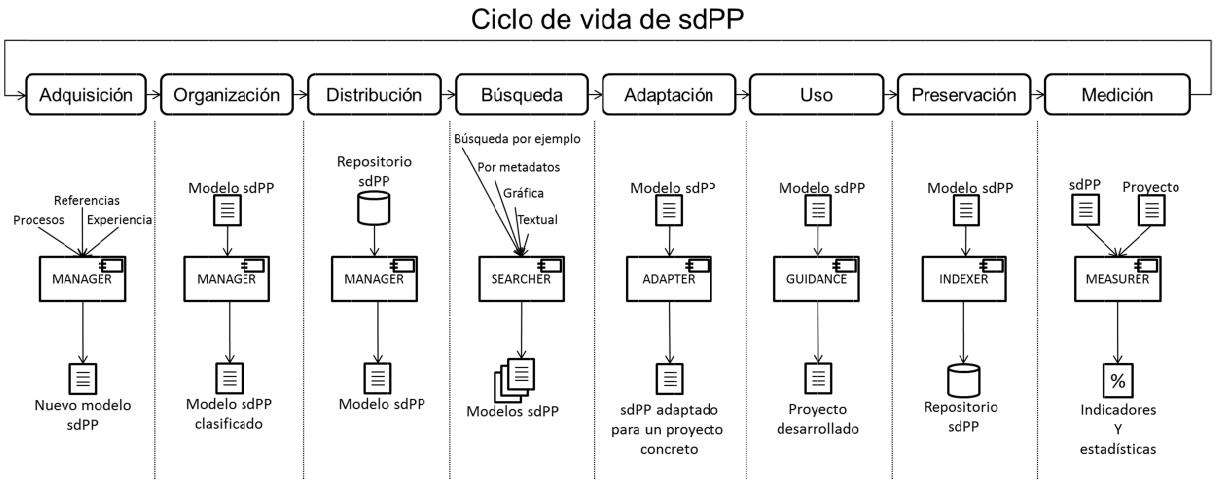


Figura 48 Ciclo de vida de un sdPP y la funcionalidad que ofrece sdpReuser

sdpReuser es una herramienta desarrollada en C# con Visual Studio 2010. El objetivo de esta herramienta es ofrecer cobertura en la gestión de sdPPs durante el ciclo de vida del conocimiento basado en patrones de proyecto.

VALIDACIÓN

Tabla de contenidos: Validación

4: VALIDACIÓN

4.1 Introducción

Como fruto de la investigación realizada en esta parte de la presente tesis doctoral se han publicado los siguientes artículos de revista indexados en el JCR: (García et al., 2012), (Martín et al., 2012a), (Martín et al., 2012b).

El objetivo de la fase de validación consiste en verificar que el framework *sdpFramework*, propuesto en la presente tesis doctoral produce mejoras en todas las fases de la gestión de conocimiento y en el ciclo de vida de los patrones; especialmente en las fases de:

- **Formalización**, favoreciendo la elicitación de conocimiento.
- **Adaptación**, facilitando que un patrón de proyecto se ajuste a las necesidades concretas de un entorno de desarrollo.
- **Utilización**, ayudando a la ejecución del desarrollo software mejorando la calidad del producto software.

La hipótesis de investigación de esta tesis doctoral y que se pretende validar en este capítulo es:

> *Si se ofrece un framework integral compuesto por un modelo de conocimiento para patrones de proyecto, un conjunto de procesos (aplicable desde la creación hasta el uso de los patrones de proyecto) y una herramienta para su gestión; entonces en las organizaciones de desarrollo de software:*
>
> - *Se facilitará la formalización de buenas prácticas de ingeniería de software aplicables a un proyecto de desarrollo, identificando los factores críticos para la elicitación del conocimiento.*
> - *Se mejorará la calidad del producto software sin afectar significativamente en el tiempo de desarrollo.*
> - *Se facilitará la identificación y aplicación de los elementos de conocimiento que ayudan a la adopción de las prácticas efectivas en las fases de adquisición, adaptación y utilización.*

A fin de hacer más simple el proceso de validación de la hipótesis se dividirá en las siguientes sub-hipótesis:

Sub-Hipótesis H1: Si se ofrece un framework integral compuesto por un modelo de conocimiento para patrones de proyecto que contenga los principales elementos de información referentes a proyectos de desarrollo software que se muestran en las metodologías más utilizadas, marcos de referencia, buenas prácticas, lecciones aprendidas y conocimiento tácito de organizaciones de desarrollo de software; entonces se facilitará la formalización de patrones aplicables a un proyecto de desarrollo. Demostrar que sdPP ayuda a formalizar las buenas prácticas aplicables a un proyecto de desarrollo, identificando los factores que afectan a su correcta formalización.

Sub-Hipótesis H2: Si se ofrece un framework integral compuesto por una herramienta informática para la gestión de patrones de proyecto se mejorará la calidad de los productos de desarrollo software sin afectar de forma significativa al esfuerzo en el desarrollo del software.

Sub-Hipótesis H3: Si se ofrece un framework integral compuesto por un conjunto de procesos para la gestión de patrones de proyectos de desarrollo software que especifique cada una de las fases del ciclo de vida del conocimiento referente a los patrones de proyecto; se facilitará la identificación de los elementos que ayudan a una correcta aplicación de las buenas prácticas, y además se ayudará a saber cuándo y cómo aplicar un sdPP.

Para poder validar la hipótesis y las sub-hipótesis según las directrices descritas en el apartado 1.6 (Método de investigación), se ha diseñado un caso de **caso de estudio embebido** compuesto por dos partes formado por un **estudio empírico y** un **experimento** en donde se aplica el framework sdpFramework. Un caso de estudio embebido es un caso de estudio que contiene más de una sub-unidad de análisis (Yin, 2009). Al igual que un caso de estudio, la metodología de un caso de estudio embebido proporciona un medio para la integración de métodos cuantitativos y cualitativos en un estudio de investigación individual (Scholz & Tietje, 2002), (Yin, 2009). Sin embargo, la identificación de las sub-unidades de investigación permite un nivel más detallado de la investigación. Se ha decidido realizar un caso de estudio embebido en la experimentación de esta tesis ya que la experimentación se va a dividir en dos partes y se van a contrastar evidencias entre ellas. Un estudio empírico es una forma de adquirir conocimiento mediante la observación directa e indirecta o la experiencia, en la primera parte de la experimentación presentada en esta tesis nos va a ayudar a obtener cierto conocimiento basado en la observación y que nos ayudará a saber ciertas evidencias empíricas del modelo sdPP. La evidencia empírica (el registro de las propias observaciones directas o experiencias) puede ser analizada cuantitativamente o cualitativamente. Un experimento es un procedimiento mediante el cual se trata de comprobar (confirmar o verificar) una o barias hipótesis relacionadas con un

determinado fenómeno, mediante la manipulación y el estudio de las correlaciones de las variables que presumiblemente son su causa. Esté método de investigación nos ayudará a validar las siguientes hipótesis:

- La aplicación del marco sdpFramework consigue una mejora en la calidad de los productos de desarrollo software
- La aplicación del marco sdpFramework consigue una mejora en el esfuerzo empleado en un desarrollo software.

Con el caso de estudio embebido diseñado para validar la hipótesis propuesta en esta tesis doctoral, se consigue una gran robustez en los resultados (Yin, 2009), además se usarán un estudio empírico y un experimento que asegurarán la validez y robustez de los resultados que permitirán verificar la validez de la hipótesis y sub-hipótesis planteadas en esta tesis doctoral.

4.2 Definición y diseño del caso de estudio embebido

Como se ha descrito anteriormente el caso de estudio embebido se ha dividido en dos partes. Cada una de las partes tiene objetivos distintos y analizan distintos aspectos para poder validar la hipótesis y sub-hipótesis. Las dos partes del caso de estudio embebido son consecutivas en el tiempo. La primera parte "Modelado de conocimiento en sdPPs" se producen cuatro tipos de sdPPs que se utilizarán en la segunda parte "Utilización del conocimiento de un sdPP" tal y como se describen en Figura 49.

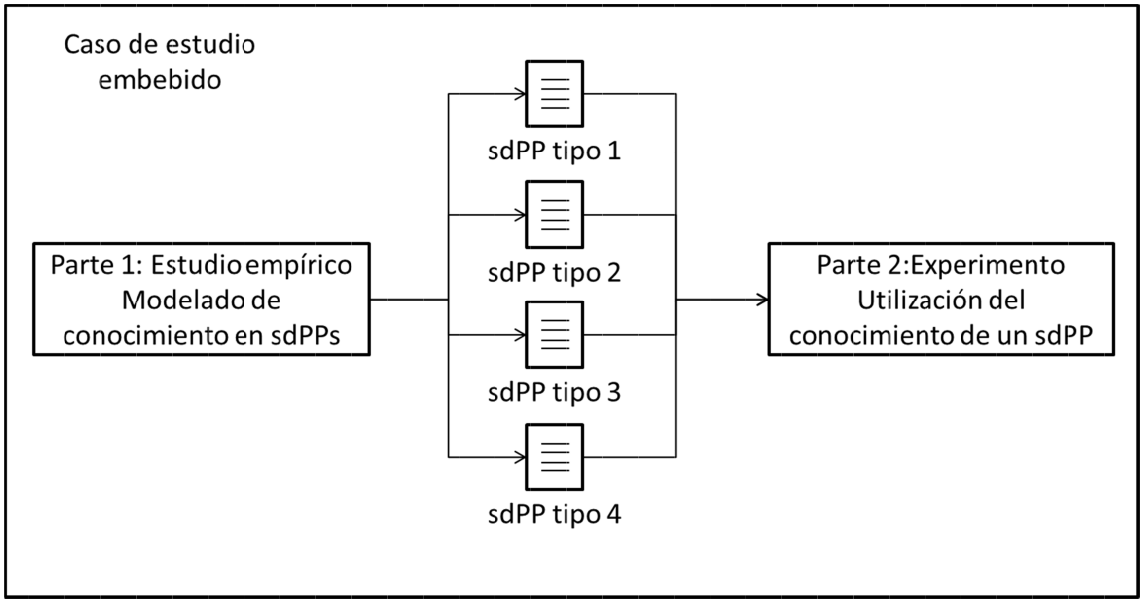


Figura 49 Descripción gráfica del caso de estudio embebido

El objetivo del estudio empírico realizado en la primera parte (Modelado de conocimiento en sdPPs) es analizar los factores que influencian en la correcta elicitación y formalización de patrones de proyecto en forma de sdPPs. Además se estudiará la utilidad de los elementos de conocimiento propuestos por el modelo sdPP para ayudar en el desarrollo de proyectos software.

El objetivo del experimento de la segunda parte (Utilización del conocimiento de un sdPP para el desarrollo de un proyecto software) consiste en analizar si el uso de sdPP ofrece una mejora en la calidad con respecto a no utilizar ningún otro sistema de gestión del conocimiento, estudiar si esto supone un incremento estadísticamente significativo del esfuerzo. Además se estudiará la percepción de la utilidad de los elementos que componen el modelo sdPP en las distintas fases del ciclo de vida propuesto.

Se han formulado varias preguntas de investigación que sirvieron para diseñar, planificar y guiar la experimentación. Estas fueron:

1. ¿Cuáles son los factores que influencian el correcto modelado de sdPPs? Para centrar la recogida de datos y su evaluación, se han tenido en cuenta varios factores, entre ellos: la colaboración entre los ingenieros de software: el conocimiento usado pre-existente y la experiencia previa en el modelado de procesos. Esta pregunta de investigación está relacionada directamente con la sub-hipótesis H1, y se tratará de validar en la primera parte de la experimentación (Modelado de conocimiento en sdPPs).
2. ¿La aplicación del conocimiento de un sdPP aumenta la calidad de los productos de desarrollo software? Esta pregunta de investigación está directamente relacionada con la sub-hipótesis H2 y se validará en la segunda parte de la experimentación: "Utilización del conocimiento de un sdPP"
3. ¿La aplicación de las actividades para aplicar el conocimiento de un sdPP aumenta el esfuerzo en el desarrollo de un proyecto software? Esta pregunta de investigación está directamente relacionada con la sub-hipótesis H2, y se validará en la segunda parte de la experimentación: "Utilización del conocimiento de un sdPP"
4. ¿Cuál es la utilidad de los elementos de información proporcionados por sdPP a través de las fases del ciclo de vida de los patrones? Se estudiará la percepción de la utilidad de cada uno de los elementos de conocimiento de un sdPP en las distintas fases en las que se ha realizado la experimentación, es decir, en el modelado, adaptación y uso del conocimiento. Esta pregunta de

investigación está directamente relacionada con la sub-hipótesis H3. Esta pregunta de investigación se validará en las dos partes de la experimentación, ya que se analizará la utilidad de los elementos de conocimiento de sdPP para cada una de las fases estudiadas en la experimentación.

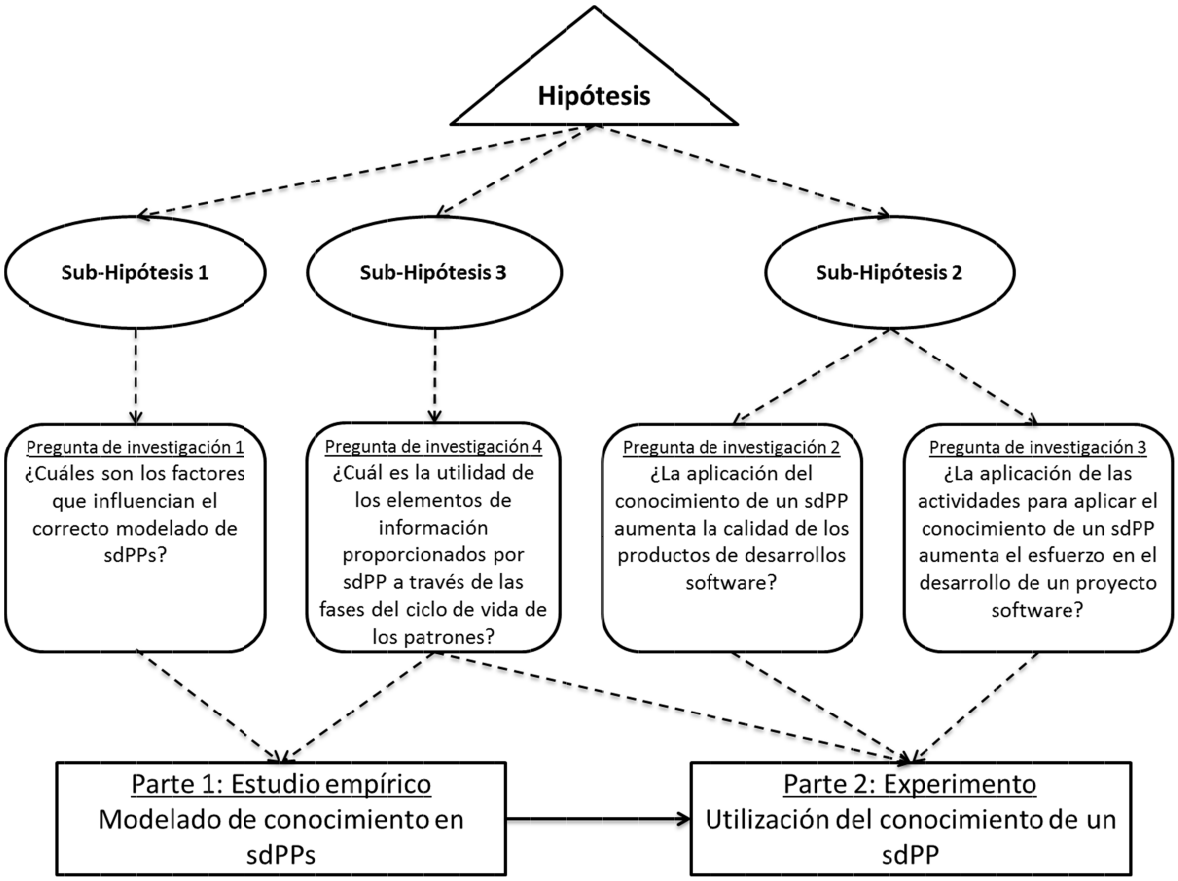


Figura 50 Correspondencia entre sub-hipótesis, preguntas de investigación y partes experimentales

Cada pregunta de investigación se tratará de responder con cada una de las fases del experimento, y cada una de estas preguntas de investigación estará relacionada con cada una de las sub-hipótesis. En la Figura 50 podemos ver la correspondencia entre las fases experimentales y las preguntas de investigación, y éstas con cada una de las sub-hipótesis.

4.2.1 Contexto

En esta sección se describirá el contexto del caso de estudio embebido. Primero se describirá el contexto de la primera parte de la investigación:

"Modelado de conocimiento en sdPPs" y posteriormente se describirá el contexto de la segunda parte: "Utilización del conocimiento de un sdPP"

A. Definición del contexto del estudio empírico (Modelado de conocimiento en sdPPs)

Doce ingenieros de software junior en la Universidad Carlos III de Madrid llevaron a cabo un estudio empírico. Estos ingenieros junior de software (a partir de ahora participantes) fueron estudiantes de último curso de ingeniería informática con un amplio conocimiento en métodos de ingeniería del software y herramientas, además tienen experiencia profesional en organizaciones de desarrollo software. Los participantes colaboraron con el grupo de investigación Knowledge Reuse en la universidad Carlos III de Madrid para crear patrones de proyecto. Los participantes en el caso empírico tenían que modelar cuatro tipos de patrones, especialmente orientados en los distintos tipos de sistemas de software. Esos cuatro tipos fueron:

Tipo 1: Un patrón de proyecto para desarrollar webs pequeñas y medianas, además de sistemas móviles para pequeños equipos, donde los requisitos cambian y varias mejoras deben entregarse de acuerdo a un calendario establecido.

Tipo 2: Un patrón de proyecto para desarrollar componentes software pequeño y mediano, y equipos de desarrollo pequeños, centrándose en la calidad del diseño, la ejecución de las pruebas e integración continúa.

Tipo 3: Un patrón de proyecto para desarrollar componentes software para sistemas de información de grandes empresas, aplicando un ciclo de vida iterativo e incremental.

Tipo 4: Un patrón de proyecto para desarrollar componentes software refinados integrando y adaptando componentes de software ya existentes.

Se ha realizado un diseño ortogonal y fraccionado donde cada uno de los doce participantes modeló tres de los cuatro tipos de patrones. En esta fase se obtuvieron 36 modelos sdPP (ver Tabla 9)

Tipo de patrón	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
sdPP tipo 1	Si	Si	Si	No	Si	Si	Si	No	Si	Si	Si	No
sdPP tipo 2	Si	Si	No	Si	Si	Si	No	Si	Si	Si	No	Si
sdPP tipo 3	Si	No	Si	Si	Si	No	Si	Si	Si	No	Si	Si
sdPP tipo 4	No	Si	Si	Si	No	Si	Si	Si	No	Si	Si	Si

Tabla 9 Asignaciones de los tipos de patrones a los participantes en el estudio empírico (parte 1: Modelado de conocimiento en sdPPs).

Para evaluar la utilidad de cada uno de los elementos de información de los patrones de proyecto, doce ingenieros de software diferentes, con un perfil similar a los participantes que crearon el sdPP concreto, evaluaron la efectividad global de los sdPP creados anteriormente.

B. Definición del contexto del experimento (Utilización del conocimiento de un sdPP)

El objetivo de la validación experimental consistió en la adaptación de sdPPs específicos y su uso para desarrollar una aplicación pequeña de software. Se ha realizado un experimento ortogonal y fraccionado, donde cincuenta participantes fueron divididos en dos grupos: el de test y el de control (ver Tabla 10). El grupo de control se compuso por dieciséis equipos (formados por dos personas cada equipo) y tuvieron que desarrollar una pequeña aplicación de software con la ayuda de un sdPP concreto cómo ayuda para su desarrollo. El grupo de control, compuesto por nueve equipos (formados por dos personas cada equipo), no usaron ningún tipo de sdPP para el desarrollo.

	Tipo sdPP	**#Grupos**	**Grupos total**
Grupos de test	sdPP tipo 1	4 (8 personas)	16 (32 personas)
	sdPP tipo 2	3 (6 personas)	
	sdPP tipo 3	4 (8 personas)	
	sdPP tipo 4	5 (10 personas)	
Grupos de control	No sdPP	9 (18 personas)	9 (18 personas)

Tabla 10 Asignación de sdPPs en el experimento (parte 2: Utilización del conocimiento de un sdPP)

Los investigadores consideraron el grupo de participantes como representativo debido a que su educación, la experiencia previa y el entrenamiento recibido, es similar a los profesionales de la industria; teniendo los mismos perfiles que el personal que trabaja en empresas innovadoras del área de las IT, que tienen poca experiencia en procesos de desarrollo de software y modelado de la información.

Tanto el estudio empírico como el experimento fueron planeados, guiados, controlados y evaluados por dos miembros del grupo de Knowledge Reuse con más de diez años de experiencia en ingeniería del software y mejora de procesos del software.

4.2.2 Plan

La planificación del caso de estudio embebido se ha divido en dos partes como hemos visto en la Figura 49, primero se describirá el plan para la primera parte: estudio empírico (Modelado de conocimiento en sdPPs) y posteriormente se describirá el plan de la segunda parte: experimento (Modelado de conocimiento en sdPPs)

A. Plan del estudio empírico (Modelado de conocimiento en sdPPs)

Como vemos en la Figura 51, la ejecución del estudio empírico se dividió en tres fases.

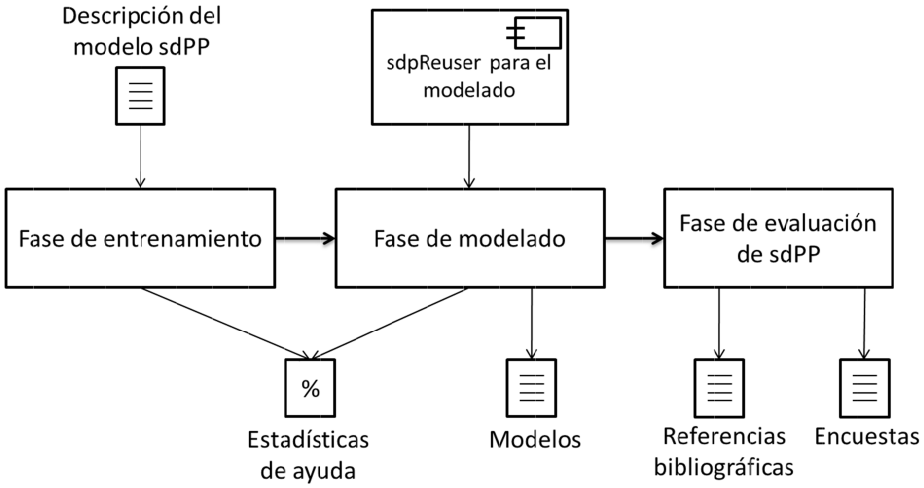


Figura 51 Plan del caso empírico

1. La primera fase del estudio empírico consistió en el entrenamiento de los participantes en los del modelo sdPP, el modelado de procesos con sdPP y el uso de la herramienta sdpReuser para la creación de patrones proyectos de desarrollo software. Las sesiones de entrenamiento consintieron en 8 horas de teoría al principio de la ejecución estudio empírico y entrevistas informales con los participantes para resolver problemas concretos en la elaboración de los sdPPs.
2. Durante la fase de modelado, cada participante modeló los tres sdPPs que se les ha asignado ayudado por la herramienta sdpReuser. Los expertos supervisaron a los participantes, las tareas realizadas durante esta fase fueron:
    a. Identificación de las referencias (libros, artículos, paginas web, definiciones de procesos prexistentes) necesarias para

ayudar la creación de un sdPP. Los expertos en esta fase no recomendaron ninguna referencia, los participantes las seleccionaron libremente. Esto es debido al hecho de que en contextos reales el personal a cargo del modelado de procesos no tienen este tipo de guías.

b. Identificación de la parte de la descripción del problema del patrón. El propósito de esta tarea fue categorizar los tipos de proyectos que podían ser desarrollados, según el modelo de sdPP descrito en la sección 3.2 (El modelo de conocimiento sdPP)

c. Modelado de la parte de la descripción de la solución del patrón. El propósito de esta tarea fue formalizar y describir los elementos de conocimientos para modelar apropiadamente el sdPP que se está formalizando, siguiendo el modelo de sdPP descrito en la sección 3.2 (El modelo de conocimiento sdPP)

Durante la fase de ejecución, los expertos han tenido reuniones informales e individuales con los participantes para resolver problemas específicos sobre el modelado del tipo de sdPP asignado. En estas reuniones, los problemas discutidos se referían a las dificultades específicas en entender los elementos de información del modelo sdPP, la implementación de las actividades requeridas para crear un sdPP y la formalización del tipo de conocimiento relevante para cada tipo de sdPP. Estas entrevistas informales no fueron planeadas previamente, si no que fueron requeridas por los participantes para resolver problemas específicos con la implementación de sdPPs

En los anexos de esta tesis doctoral se ofrece un ejemplo de sdPP, aunque existen varios ejemplos de sdPPs en formato web creado por los participantes en la siguiente dirección web:

http://www.diego-martin.info/sdPP-example

3. En la fase de evaluación de los sdPPs generados se realizaron las siguientes tareas:

a. Los sdPPs creados se presentaron a otros doce ingenieros de software distintos a la fase anterior los cuales tuvieron que usarlos para desarrollar un proyecto de software en el grupo de Knowledge Reuse de la Universidad Carlos III de Madrid. Ellos evaluaron la efectividad global de los sdPPs obtenidos en la fase anterior a través de unas encuestas

donde se preguntaba por sus impresiones en la utilidad del modelo sdPP, así como, la relevancia, satisfacción, facilidad de uso y frustración. Además, los coordinadores del estudio empírico analizaron la relevancia de las referencias usadas por los participantes para ayudar en la creación de los sdPPs.

b. Los expertos anteriormente mencionados evaluaron los patrones generados en la fase de modelado. Cada experto evaluó con una puntuación a cada uno de estos patrones según lo correcto de cada modelo. En la última fase, se les preguntó a los participantes por el conjunto de fuentes bibliográficas usadas por cada sdPP que ellos modelaron. Los mismos expertos juzgaron cada referencia bibliográfica usada para determinar su grado de relevancia para el modelado de patrones de patrones.

A. Plan del experimento (Utilización del conocimiento de un sdPP)

Se quiere validar las fases de adaptación y de utilización de conocimiento propuesto en el ciclo de vida de los sdPPs, para ello se ha diseñado un experimento en tres fases, tal y como muestra la Figura 52.

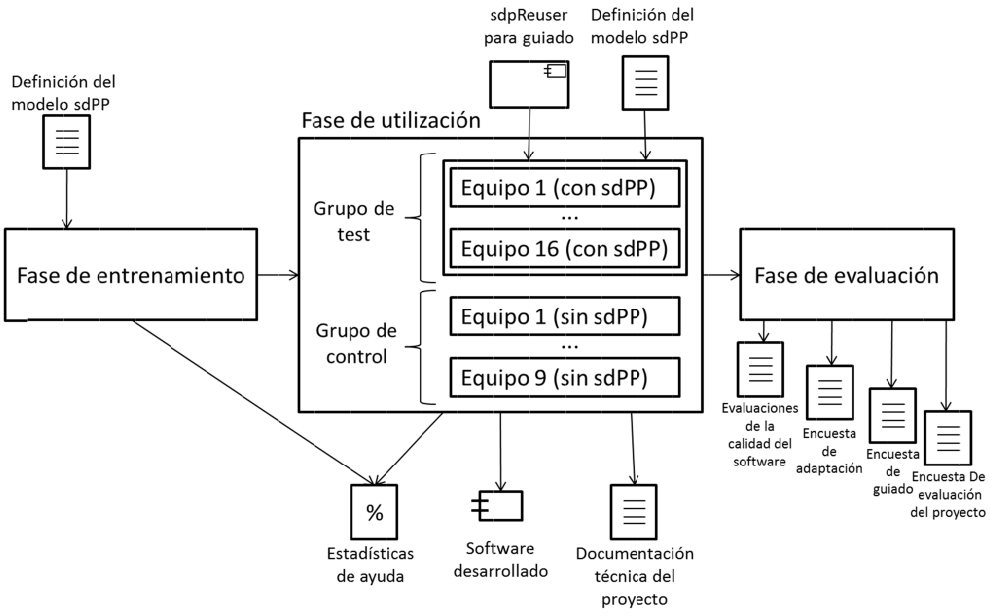


Figura 52 Plan de la experimentación

- **Fase de entrenamiento sobre el sdPP:** El objetivo de esta fase fue entrenar a los ingenieros de software en el uso de sdPPs durante el desarrollo de un proyecto de software. Los participantes

recibieron doce horas de entrenamiento incluyendo la presentación del modelo de sdPP y sus elementos de conocimiento, sesiones prácticas y teóricas sobre la funcionalidad de sdpReuser y su sistema de guiado para guiar en el desarrollo de software.

- **Fase de utilización del sdPP (desarrollo)**: Durante esta etapa cada grupo tubo que desarrollar uno de las dos aplicaciones asignadas por los expertos.

  La primera aplicación fue un portal web para la creación automática de documentación HTML a partir de una fuente de datos externa procedente de otro sistema responsable en la gestión de la información de modelos UML; estos datos procedían de una base de datos de una aplicación externa al proyecto. Posteriormente, el usuario puede modificar partes del documento a través de un editor web de HTML. El sistema tiene dos tipos de usuarios: administradores y usuarios normales. Loa administradores eligen las fuentes de datos mientras que los usuarios pueden elegir la fuente de datos para generar la documentación y modificar los documentos HTML.

  La segunda aplicación software consintió en una aplicación web para la gestión de juicios de relevancia de documentos HTML. El sistema tiene dos tipos de usuarios: administradores y evaluadores. El sistema controla el acceso a la web, utilizando un elemento de inicio de sesión. Los administradores gestionan los documentos que tienen que ser evaluados, las opciones de evaluación, asignando tareas de evaluación a cada evaluador; además puede ver los resultados ofrecidos por los evaluadores. Los evaluadores juzgan cada documento marcado por los administradores y ofrecen su juicio de relevancia a través de la web. La Figura 53 muestra una captura de pantalla de una aplicación realizada por un grupo de desarrollo.

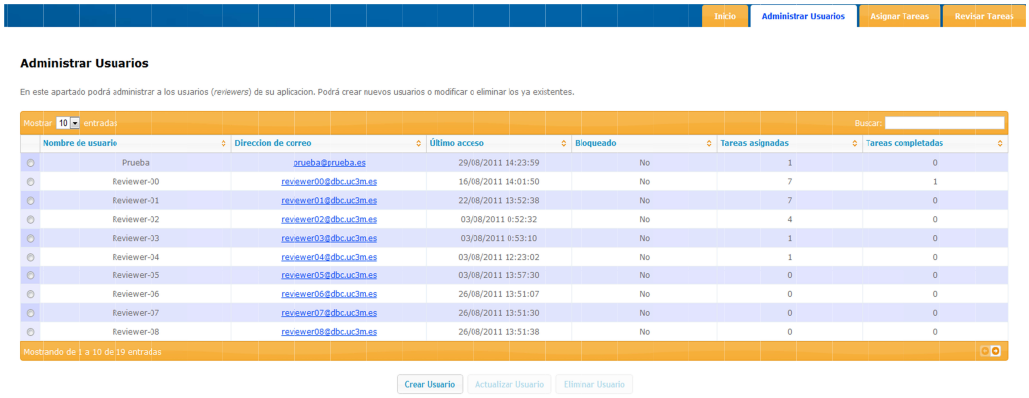


Figura 53 Captura de pantalla de una aplicación desarrollada

La aplicación del sdpFramework durante la fase de uso se dividió en dos pasos:

- Adaptación: este paso consistió en la búsqueda del sdPP más apropiado para un proyecto concreto y un plan de proyecto incluyendo actividades, productos y "to-dos" obtenidos a través de la instanciación del sdPP seleccionado.
- Aplicación (o uso): En este paso el sdPP adaptado se utilizó como guía en las actividades del desarrollo de software.

  Durante esta etapa, los expertos ayudaron en reuniones informales con los miembros del equipo de desarrollo para resolver problemas específicos a la hora de buscar y adaptar los sdPP más recomendados para el proyecto de software. En estas reuniones, las discusiones relacionadas con las dificultades en entender los elementos de información incluidos en cada uno de los sdPP y el uso de la herramienta sdpReuser para adatar un sdPP concreto un proyecto concreto de desarrollo software. Estas reuniones no fueron planeadas previamente, sin embargo, fueron requeridas por los desarrolladores.
- Fase de evaluación del sdPP: Todos los participantes de esta validación experimental rellenaron varias encuestas de evaluación. Los participantes en el grupo de test tuvieron que completar una encuesta especifica sobre la adaptación y el uso del sdPP durante el desarrollo. Mientras que los participantes del grupo de control tuvieron que rellenar una encuesta genérica sobre las dificultades en la ejecución del proyecto de desarrollo asignado.

4.2.3 Recopilación de la información

Los datos recopilados para responder la pregunta de investigación 1 (¿Cuáles son los factores que influencian el correcto modelado de sdPPs?) fueron:

a) Información sobre la corrección de los sdPPs creados por los participantes en el estudio empírico. Los expertos que gestionaron el estudio empírico evaluaron cada patrón modelado, juzgando la corrección de los elementos de conocimiento de cada sdPP según los criterios de la Tabla 11. Los expertos que ejecutaron el estudio empírico siempre tuvieron en mente la definición del modelo de información de sdPP. El global de la corrección de los patrones creados es un agregado de las puntaciones obtenidas por cada uno de los criterios. El valor final de la corrección se calculó como la media entre la evaluación proporcionada por los dos expertos.

Elemento sdPP	Cuestiones analizadas	Peso
Descripción	¿Es la descripción precisa?	8
	¿Es el tamaño de la descripción correcto?	2
Workflow	¿Han sido las actividades modeladas como tal? (Identificación conceptual de las actividades)	6
	¿Son las actividades apropiadas para el tipo de proyecto?	7
	¿Tienen las actividades el nivel correcto de abstracción?	2
	¿Es la secuencia del workflow adecuada para el tipo de proyecto?	15
	¿La semántica del workflow ha sido usada correctamente?	10
Productflow	¿Han sido los productos identificados correctamente como tal?	8
	¿Han sido los productos correctamente identificados?	7
	¿Han sido las dependencias entre los productos correctamente identificadas?	5
To-dos	¿Han sido los conjuntos de "to-dos" y los de "not to-dos" identificados adecuadamente?	10
	¿Se han confundido con otros elementos del modelo?	5
Requisitos	¿Se han identificado algún requisito? ¿Son relevantes?	3
Riesgos	¿Se han identificado algún riesgo? ¿Son relevantes?	5
Metadatos	¿Se han identificado algún metadato? ¿Son relevantes?	5

Tabla 11 Preguntas para analizar la corrección un sdPP

b) Información sobre la relevancia de las referencias bibliográficas que los participantes utilizaron para modelar los sdPPs. Los participantes seleccionaron estas referencias libremente, sin la intervención de los expertos que guiaron el estudio empírico. Los expertos que gestionaron el estudio empírico evaluaron la relevancia de las referencias utilidades por cada sdPP creado bajo los criterios que se muestran en la Tabla 12 definida por los directores de estudio empírico.

Elemento sdPP	Cuestiones analizadas	Peso
Descripción	¿Proporciona la bibliografía información relevante para la descripción?	1
Workflow	¿Proporciona la bibliografía información relevante para el workflow?	1
Productflow	¿Proporciona la bibliografía información relevante para el productflow?	1
To-dos	¿Proporciona la bibliografía información relevante para los "to-dos"?	1
Metadatos	¿Proporciona la bibliografía información relevante para los metadatos?	1
Requisitos	¿Proporciona la bibliografía información relevante para los requisitos?	1
Riesgos	¿Proporciona la bibliografía información relevante para los riesgos?	1

Tabla 12 Criterios de evaluación de las fuentes bibliográficas que ayudaron a la creación de cada sdPP

c) Información sobre la experiencia previa de los ingenieros de software que participaron en el estudio empírico. Se ha preguntado a los participantes por los años de experiencia desarrollando componentes similares de software, número de asignaturas sobre procesos de software recibidas en la carrera y la experiencia previa en las prácticas efectivas que se pueden incluir en cada patrón de proyectos. Esta información fue recopilada a través de un cuestionario que rellenaron cada uno de los participantes al final de la ejecución del estudio empírico.

d) Finalmente, los expertos recopilaron información sobre la colaboración y el conocimiento compartido entre los creadores de los patrones. Los datos específicos recopilados bajo esta categoría fueron el número y tiempo que ellos emplearon en las reuniones individuales con los participantes para clarificar dudas y problemas sobre cómo modelar sdPPs o las prácticas efectivas a ser consideradas por su inclusión en un patrón de proyectos.

Se va a utilizar estadística descriptiva, pruebas de correlación y análisis de regresión entre estos datos para poder discutir la pregunta de investigación 1.

La información numérica previamente descrita para la pregunta de investigación 1 fue completada con información obtenida de las observaciones de los expertos obtenidas como consecuencia de entrevistas informales llevadas a cabo con los participantes. Esta información fue utilizada para identificar aquellas correlaciones y regresiones obtenidas del análisis estadístico demuestran que no han sido casuales y que son coherentes con la información obtenida de los participantes del estudio empírico. En este sentido, la información obtenida de las entrevistas y las observaciones asociadas fueron útiles para determinar los resultados estadísticos relevantes para su análisis.

La información recopilada para afrontar la segunda pregunta de investigación (¿La aplicación del conocimiento de un sdPP aumenta la calidad de los productos de desarrollos software?) es:

a) La evaluación de la calidad de los proyectos que los participantes han desarrollado. Dos expertos evaluaron todos los productos desarrollados por cada grupo de desarrollo usando los criterios de la Tabla 13.

Elemento del proyecto	**Cuestiones analizadas**		**Peso**
Requisitos	Proyecto de software 1	Proyecto de software 2	
	La aplicación debe seguir una arquitectura cliente servidor		1
	La aplicación debe asegurar el control de acceso a los usuarios autorizados		
	La aplicación debe proveer dos tipos de usuarios: Administradores y usuarios		
	Los administradores deben ser responsables de la gestión de usuarios.		
	Los administradores seleccionan la fuente de datos de una aplicación externa.		
	La estructura de los datos externa es un modelo UML que sigue el formato XMI.	Una tarea es un conjunto de HTMLs que deben ser juzgados por los usuarios.	
	Los usuarios generan informes en HTML automáticamente seleccionando partes del XMI que ellos consideren relevantes del modelo en UML	Los administradores crean tareas que los usuarios deben ejecutar	
	Los usuarios pueden modificar los informes manualmente usando un editor HTML	Los administradores asignan tareas a los usuarios	
	Los informes se debe almacenar en una base de datos	Los usuarios evalúan cada documento dentro de una tarea y para un "topic" concreto. Cada evaluación se llama juicio de relevancia.	
	Los informes deben tener al menos un título, una cabecera y un píe-	Los juicios de relevancia deben guardarse en una base de datos	
	Opcionalmente, los informes puede tener uno o mucho elementos "body"	Un "topic" es una pregunta sobre un documento y un conjunto de valores de relevancia	
	Los administradores pueden pre visualizar los informes	Los administradores pueden comprobar la completitud de cada una de las tareas	
Atributos del software desarrollado	¿Hay una correcta encapsulación de los componentes?		2
	¿Hay una correcta definición de las interfaces?		
	¿Se ha definido test de integración?		1
	¿Se han usado patrones de diseño correctamente?		
Documentación	¿Se ha desarrollado correctamente la documentación técnica del proyecto?		2

Tabla 13 Criterios de evaluación de los proyectos desarrollados

La información recopilada para afrontar la tercera pregunta de investigación (¿La aplicación de las actividades para aplicar el conocimiento de un sdPP aumenta el esfuerzo en el desarrollo de un proyecto software?), consistió en el esfuerzo empleado en el desarrollo de todas las aplicaciones software hechas por el grupo de test y el de control. Se midieron los tiempos de desarrollo de distintas actividades, así como el tiempo necesario para aprender

a usar el conocimiento de un sdPP y la implementación de las actividades de gestión de un sdPP para los grupos de test, mientras que para los grupos de control además de medir el tiempo de desarrollo de software se midió el tiempo de aprendizaje de una metodología de desarrollo si decidieron utilizar alguna.

Los datos recopilados para responder a la cuarta pregunta de investigación (¿Cuál es la utilidad de los elementos de información proporcionados por sdPP a través de las fases del ciclo de vida de los patrones?) fueron las evaluaciones subjetivas sobre la usabilidad de doce ingenieros de software en la primera parte del diseño experimental ("Modelado de conocimiento en sdPPs") y las evaluaciones subjetivas de 32 personas en la segunda parte del experimento (Utilización del conocimiento de un sdPP). Estos criterios de usabilidad fueron adaptados de (Nodder & Nielsen, 2008) para determinar la utilidad, relevancia, satisfacción y frustración para interpretar cada elemento de conocimiento incluidos en el modelo sdPP. Esta información sobre la usabilidad se obtuvo a través de una encuesta anónima completada por los participantes al final de la ejecución de las partes experimentales.

4.3 Resultados

Esta sección presenta los resultados obtenidos durante la ejecución del caso de estudio embebido desarrollado en esta tesis.

Los resultados del caso de estudio embebido se presentarán de forma estructurada dando respuesta a las respuestas de investigación.

4.3.1 Factores que afectan en el correcto modelado de sdPPs

Para poder determinar los factores que afectan a un correcto modelado de los sdPPs, inicialmente se analizó la corrección de lo patrones de proyecto producidos por el estudio empírico por los participantes. Luego, se trató de identificar los factores que contribuyeron al modelado correcto de patrones buscando correlaciones entre los factores y usando técnicas de prueba de hipótesis entre los variables del experimento como la relevancia de las fuentes bibliográficas que usaron los participantes, la información sobre la experiencia previa de los ingenieros de software que participaron en el estudio empírico, y el esfuerzo empleado en las sesiones para discutir con los expertos.

La Tabla 14 muestra un resumen de cuanto ha sido de correctos los patrones modelados por tipo de patrón. La media de la corrección de los patrones es 3,39 sobre 5. El valor mínimo observado fue 2,23 para un patrón del tipo 3, mientras que el valor máximo fue 4.43 para un participante distinto al

anterior que modelo el tipo de proyecto 4. En general, el 75% de las evaluaciones estuvieron entre 3 y 4.

	sdPP Tipo 1	sdPP Tipo 2	sdPP Tipo 3	sdPP Tipo 4	Media
1. Corrección del modelo	3.53	3.43	3.11	3.51	3.39
1.1 Descripción	3.59	3.96	3.14	4.01	3.67
1.2 Workflow	3.87	3.48	3.43	3.76	3.63
1.3 Productflow	3.65	3.86	3.39	4.28	3.79
1.4 To-dos	2.94	3.07	2.52	2.20	2.68
1.5 Metadatos	1.00	1.34	1.46	1.34	1.65
1.6 Requisitos	2.31	2.24	1.40	1.72	1.92
1.7 Riesgos	2.32	1.64	1.13	1.38	1.62
2. Relevancia de las referencias	3.61	2.67	2.22	3.61	3.03
3. Discusiones sobre las prácticas efectivas	1.33	1.67	1.56	1.33	1.47
4. Discusiones para modelar cada uno de los sdPPs	0.56	1.11	0.83	0.56	0.76

Tabla 14 Resumen sobre cuanto de correcta ha sido la formalización de patrones por tipo.

Sin embargo, la puntuación sobre la corrección de los sdPPs analizados ha sido muy diversa. La Figura 54 muestra estas distribuciones. Viendo dicha figura, se pueden identificar tres grupos distintos, formados por los siguientes elementos de conocimiento:

- Descripción, workflow, productflow; los cuales tienes valores altos
- Metadatos, requisitos y riesgos; que tienen puntuaciones bajas.
- El conjunto de "To-dos" y "Not To-dos"; que tienen valores medios.

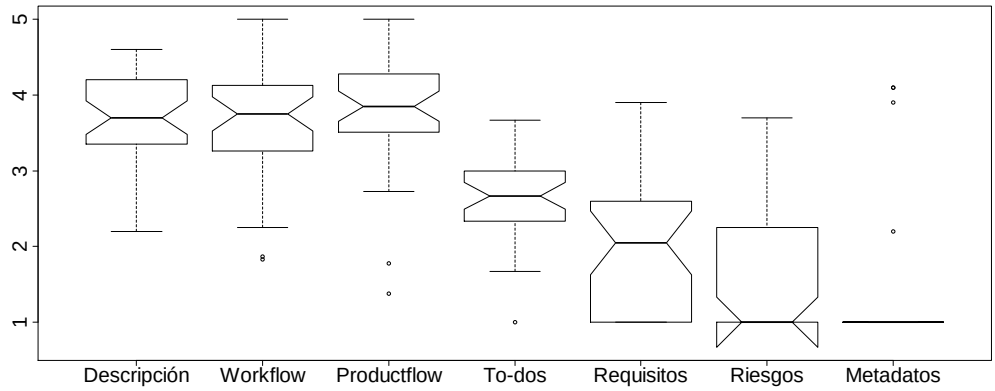


Figura 54 Distribuciones de la corrección de los sdPP por elementos de conocimiento

Se ha conseguido ajustar una regresión múltiple (R^2=0.809) entre las variables relevancia de las referencias que se usaron para formalizar el conocimiento (B), la experiencia previa en la aplicación de las prácticas efectivas de software (EM), y el grado de colaboración entre los creadores de patrones de proyecto (ER). La ecuación es:

Corrección de un patrón = 2.5406 + 0.1877 B + 0.1167 ER + 0.0936 EM

1. El primer predictor es la corrección de las referencias usadas para elicitar el conocimiento para que el patrón de proyecto sea modelado. Esto significa que un conjunto relevante de fuentes bibliográficas que provean el conocimiento explicito necesario para poder modelar un patrón de proyectos son necesarias para modelar correctamente los elementos de información de un sdPP.

   La relevancia de las referencias depende de la inclusión de información de elementos del patrón (actividades, workflow, productos y productflow). La identificación de las actividades y el workflow son los elementos que son más influenciados por la relevancia de las fuentes bibliográficas usadas para ayudar a la formalización de un sdPP. Se ha encontrado una alta correlación de Spearman (ρ =0.7064, p<0.001) entre la relevancia de las fuentes bibliográficas y el conjunto de variables que miden la corrección en la identificación de las actividades y el workflow.

   Para analizar en más detalle la importancia de este predictor, se ha realizado un estudio de la relevancia de las referencias usadas por los participantes. La Tabla 15 muestra un resumen del número de las fuentes bibliográficas por modelo y la mediana de la evaluación de su relevancia por cada conjunto de fuentes bibliográficas.

Modelo	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**
a) Número de referencias utilizadas para modelar el patrón	4	6	2	6	9	5	2	1	2	5	2	3
b) Mediana de la relevancia de cada conjunto de referencias	3.5	4	3	4.5	3	2	1.5	4	2	4	3.5	1
Modelo	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**	**P21**	**P22**	**P23**	**P24**
a) Número de referencias utilizadas para modelar el patrón	2	2	4	2	2	4	1	6	2	4	3	1
b) Mediana de la relevancia de cada conjunto de referencias	1.5	3	1	3.5	3.5	3.5	4	3.5	2	4	4	0
Modelo	**P25**	**P26**	**P27**	**P28**	**P29**	**P30**	**P31**	**P32**	**P33**	**P34**	**P35**	**P36**
a) Número de referencias utilizadas para modelar el patrón	2	2	2	7	5	2	5	4	1	8	10	5
b) Mediana de la relevancia de cada conjunto de referencias	3.5	2.5	3.5	5	3	3.5	2	3	1	4.5	4	4

Tabla 15 Resultados del análisis de las fuentes bibliográficas

La Tabla 16 muestra la existencia de una alta correlación entre la relevancia de las fuentes bibliográficas y la corrección de los sdPPs elaborados por los participantes.

Tipo de patrón	ρ entre la calidad de las fuentes bibliográficas y la calidad del modelo	ρ entre el número de Fuentes bibliográficas y la calidad del modelo
sdPP tipo 1	0.7454 (p = 0.01058)	0.8944 (p = 0.0005669)
sdPP tipo 2	0.7661 (p = 0.008032)	0.8037 (p = 0.004528)
sdPP tipo 3	0.8394 (p = 0.002327)	0.5534 (p = 0.06109)
sdPP tipo 4	0.7258 (p = 0.01343)	0.9037 (p = 0.0004148)
Total:	0.755 (p = 5.161·10-08)	0.4952 (p = 0.001067)

Tabla 16 Correlaciones entre la corrección de los sdPP y la relevancia de las fuentes bibliográficas y el número de fuentes bibliográficas agrupado por tipos.

Se puede decir la corrección de un sdPP no depende del número de las fuentes bibliográficas usadas para su modelado. Pero sin embargo, si se puede decir que un predictor para saber la corrección de un patrón es la relevancia del conjunto de fuentes bibliográficas utilizadas. Las referencias deberían incluir información sobre actividades y la identificación de productos, workflows y productflow. Otros elementos como los "to-dos", los metadatos, los requisitos y los riesgos; no suelen encontrarse fácilmente en fuentes bibliográficas públicas. Además, es importante remarcar que las formalizaciones previas de los procesos de desarrollo de las organizaciones pueden proveer la misma información que las referencias elegidas por los participantes, pero éstas son mejores porque son adaptadas a las restricciones contextuales del proyecto.

2. El segundo predictor es el grado de colaboración y discusión entre los autores de los patrones de proyecto. Muchos de los elementos de conocimiento incluidos en el meta-modelo de sdPP no son bien conocidos por los participantes. Para poder resolver este problema se ha implementado una estrategia de colaboración entre los participantes y los expertos. La colaboración entre los ingenieros de software es muy importante para poder identificar elementos clave para incluir en elementos propuestos en el modelo de sdPP.
3. El tercer predictor de la corrección de un patrón es la experiencia previa de los participantes en la aplicación de las prácticas efectivas de ingeniería software. Esta variable mide el conocimiento tácito de los ingenieros de software.

   Se ha encontrado una alta correlación (ϱ = 0,96) entre la experiencia previa en las prácticas efectivas de cada tipo de proyectos y la variable que mide cuanto de correcta ha sido la identificación de las actividades y el workflow.

Para determinar los atributos específicos que caracterizan la experiencia requerida para modelar patrones de proyecto, se ha ajustado una regresión (R^2=0.6952) entre las variables: años de experiencia en el área de organización (Yex), número de asignaturas aprobadas en la carrera relacionadas con la gestión de procesos (Nsub), y la experiencia previa en la aplicación de las prácticas efectivas del modelo de sdPP (Pex). La ecuación de la regresión múltiple es:

$$EM = 2.2715 + 0.6664\ Yex + 0.4023\ Nsub + 0.3012\ Pex$$

a) El primer predictor que tiene más influencia son los años de experiencia en la organización. Era de esperar que la experiencia previa en la organización es relevante ya que la organización de la información tiene que ver con la capacidad de abstracción necesaria para poder elicitar el conocimiento necesario de un sdPP.

b) El siguiente predictor sobre la experiencia está relacionado con las asignaturas aprobadas en la carrera de ingeniería informática relativas a la gestión de procesos de desarrollo software. Esta variable nos muestra la importancia de este tipo de asignaturas para complementar las experiencias prácticas de los ingenieros de software con información explícita en otras practicas efectivas, criterios para determinar su importancia para el desarrollo efectivo de software y como modelar este tipo de procesos en conocimiento en el ámbito de los patrones de proyectos.

c) El tercer predictor está también relacionado con la experiencia previa (profesional o educacional) en las prácticas efectivas para incluir en cada patrón de proyecto. Sin embargo, es necesario puntualizar que esta fue la variable con menos influencia en la regresión; esto es debido a que la implementación de las estrategias de colaboración entre los participantes se realizó para compensar la falta de conocimiento de los participantes en varias prácticas efectivas del modelo sdPP.

4.3.2 Análisis de la mejora de la calidad en los productos desarrollados con sdPP.

La Tabla 17 muestra un resumen de la media de la calidad de los proyectos desarrollados durante el experimento, puntuados desde 0 a 10. La evaluación de la calidad se realizó usando los criterios presentados en la Tabla 13

	Con sdPP		Sin sdPP	
	Media	SD	Media	SD
Requisitos completados	6.44	2.12	4.03	1.57
Usabilidad de la GUI	7.12	2.42	7.89	2.52
Calidad del modelo de persistencia de datos	8.56	2.16	8.89	2.67
Encapsulación de los componentes	5.94	2.17	5.89	2.89
Definición de interfaces	6.87	1.9	6.56	2.13
Calidad de las pruebas de integración	8.37	1.54	7.11	.3.65
Uso de los patrones de diseño	7	.1.5	6	2.6
Calidad de la documentación técnica	6.95	2.7	3.61	2.37
Agregado de la calidad	6.72	1.66	5.02	1.74

Tabla 17 Resumen de la calidad de los proyectos desarrollados

Los resultados obtenidos indican que los equipos que usaron sdPP mejoraron en la calidad del software. Los equipos que usaron sdPP tuvieron una media de la calidad de 6,72; mientras que los grupos que no usaron sdPP tuvieron una media en la calidad de 5,02. Se ha realizado un test de Mann-Whitney U para comprobar que hay diferencias estadísticas entre el grupo de control y el grupo de test, dicho tipo de test se usa para comprobar la heterogeneidad de dos muestras ordinales y se usará en nuestra experimentación porque el grupo de control es menor que el grupo de test. Este test permite evaluar la siguiente hipótesis el uso de sdPP como guía para el desarrollo del software mejora la calidad del software. El resultado para este test confirma la hipótesis formulada ($w = 107{,}5$, $p < .05$). Un p-valor igual o menor que 0,05 indicas que el test es significante.

Por lo tanto, el uso de sdPP contribuye a mejorar la calidad del desarrollo software para los tipos de proyectos descritos en la sección 4.2.1

Se han realizado varios análisis estadísticos para obtener una visión más clara y en profundidad sobre como sdPP ha contribuido a mejorar la calidad del desarrollo de proyectos de desarrollo. La Figura 55 muestra la distribución de las puntuaciones para los criterios analizados

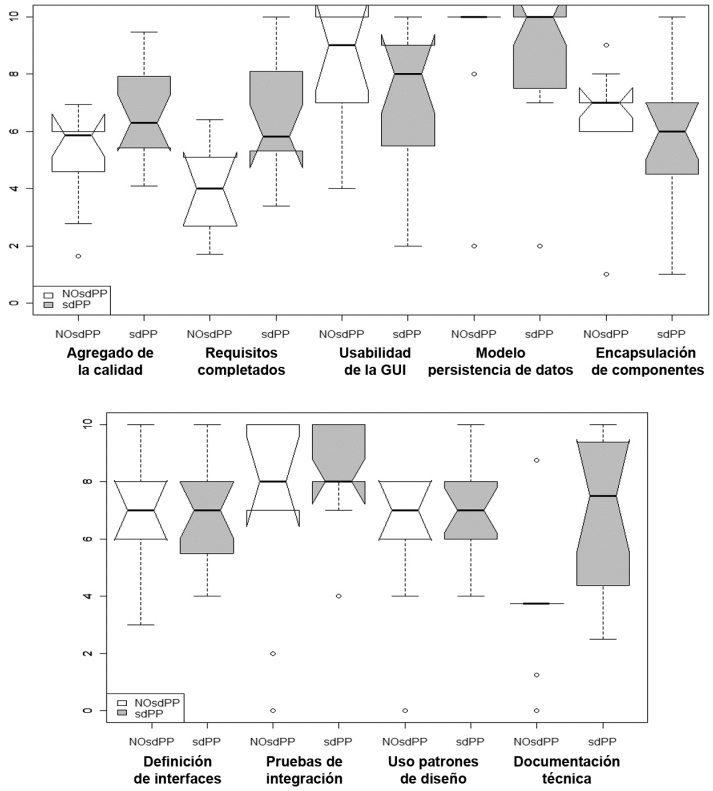


Figura 55 Diagrama de cajas de los criterios de calidad de un sdPP

El uso de sdPPs contribuyó a mejorar la calidad del software porque los equipos que usaban el framework propuesto consiguieron completar más requisitos y aumentaron la calidad de la documentación técnica del proyecto. Los grupos que usaron sdPP tuvieron una media de requisitos completados de 6,44, mientras que los grupos que no usaron sdPP tuvieron una media de 4,03. Para comprobar si hay una diferencia estadísticamente significativa entre las muestras de la variable que mide únicamente los requisitos completados realizando otro test de Mann-Whitney U. El resultado del test es positivo y confirma la hipótesis: "los equipos que han usado sdPP completan más requisitos que los equipos que no usan sdPP" ($w=118,5$, $p<0.01$), el p-valor nos indica que es significativo.

En cuanto a las funcionalidades proporcionadas en los proyectos desarrollados, la mejora con sdPP ha sido muy pronunciada. La razón de estos resultados es que los sdPP albergan las mejores prácticas de desarrollo de software, y éstas hacen hincapié en el cumplimiento de todos los requisitos. Por lo tanto, sdPP consigue que se cumplan con todos los requisitos.

Las variables que miden la "Definición de las interfaces", "Pruebas de integración" y "uso de patrones de diseño", parece que no hay diferencias entre

el uso de sdPP o no. La calidad de las "pruebas de integración" se basan en el conocimiento en el las practicas de testeo que no están incluidas en el modelo de sdPP. Los sdPPs que se han empleado no incluían no actividades específicas para la definición de interfaces o el uso de patrones de diseño, por lo tanto no se han notado diferencias entre los equipos que han usado sdPP y los que no.

Las variables que miden la calidad de la "Encapsulación en componentes", la "Usabilidad de la GUI" y del "Modelo de persistencia de datos" parece ser algo mejor en los grupos que no usaron sdPP. La diferencia en variable que mide la calidad de la "Encapsulación en componentes" puede deberse al hecho de que los especialistas que guiaron a los participantes hicieron énfasis en tener unos componentes buenos y confiables. Los grupos que no usaron sdPP tomaron esta recomendación como objetivo, mientras que los grupos que usaron sdPP siguieron las recomendaciones que proporcionaba cada patrón. La variable que mide la calidad del "Modelo de persistencia de datos" tiene unos valores muy altos para los dos grupos, aunque el grupo que uso sdPP tiene unos valores algo más bajos. La razón para estos valores es que los grupos que usaron sdPP no tuvieron suficiente tiempo para invertir en desarrollar un buen modelo de persistencia de datos ya que no era un requisito obligatorio del proyecto.

El uso de sdPPs además contribuye a la mejora de la calidad de la documentación técnica (especificación de requisitos, documentos de análisis, diseño, etc.): 6,95 contra 3,61. Se ha realizado un test de Mann-Whitney U para confirmar que los grupos que usaron sdPP obtuvieron una mejora significativa en la calidad de "documentación técnica de software". El resultado fue positivo ($W=119$, $p<0.01$); el p-valor indica que se confirma la hipótesis de forma significativa.

La razón para tener una mejora tan significativa en la calidad en la documentación técnica es porque sdPP enfatiza las actividades y prácticas que se centran en la creación de documentación técnica. Los grupos que no usaron sdPP para su desarrollo no tenían el requisito de crear ningún tipo de documentación, exceptuando por los "diagramas de componente", "diagramas de paquetes" y "documentación de diseño", estos tres documentos causó que la media de la distribución cayera ene torno al 4.

4.3.3 Análisis del esfuerzo empleado en aplicar las actividades para aplicar el conocimiento de un sdPP en un proyecto de desarrollo software.

Se ha realizado un estudio del esfuerzo necesario para el uso de sdPPs durante el proyecto de desarrollo, analizando el tiempo empleado por los equipos, tanto los que usaron sdPP y los que no lo usaron.

La Tabla 18 muestra un resumen del esfuerzo (medido en horas) requerido para aprender y adaptar los procesos y las prácticas para desarrollar software de manera independiente. Se muestran los valores de la media, mediana y desviación típica para ambos grupos.

	Con sdPP			**Sin sdPP**		
	Media	Mediana	SD	Media	Mediana	SD
Aprendizaje de sdPP	4	4	0	0	0	0
Aprendizaje de sdpReuser	6	6	0	0	0	0
Búsqueda de información sobre procesos/prácticas	0	0	0	0.67	0	1.41
Aprendizaje de procesos/prácticas	0	0	0	1.78	0	3.67
Adaptación de procesos/prácticas al proyecto específico	4.5	4.5	0.52	2.89	0	5.75
Desarrollo	46.37	48.5	9.41	47.33	48	19.11
Total	60.87	63	9.65	52.67	48	24.83

Tabla 18 Resumen del esfuerzo empleado en el desarrollo de los proyectos por grupos.

Como ya se ha indicado anteriormente, como el grupo de control no usó ningún patrón para el desarrollo, no tuvieron que emplear ningún esfuerzo para el entrenamiento sobre sdPP. Sin embargo, varios equipos del grupo de control emplearon tiempo en aprender procesos, prácticas y metodologías que les ayudaran a desarrollar el proyecto que se les asignó.

El tiempo de desarrollo para ambos grupos tiene unos valores muy similares para la media y la mediana, pero el valor de la desviación típica para los que no usaron sdPP es más del doble que para el grupo que uso sdPP (ver Tabla 18).

La Figura 56 muestra que el esfuerzo del desarrollo de los grupos que no usaron sdPP es tres veces más variable que los que usaron sdPP. Por lo tanto, el esfuerzo en el desarrollo fue menos variable en los grupos que usaron sdPP. Esto es debido a que sdPP proporciona un workflow claro y un productflow basado en metodologías de desarrollo, permitiendo a los desarrolladores tomar mayor control sobre cada una de las actividades y productos del proceso de desarrollo.

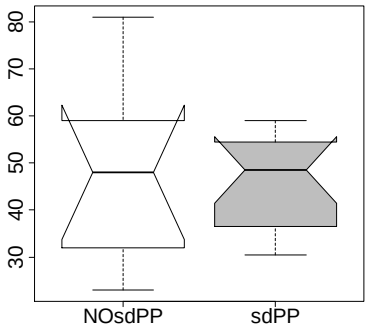


Figura 56 Diagrama de cajas para la variable "esfuerzo en el desarrollo" por grupos

La distribución de la variable "esfuerzo total", hay diferencias en la media y la desviación típica. La Tabla 18 y la Figura 57 muestran que, en términos de la media, los grupos que usaron sdPP tardaron más tiempo en desarrollar el proyecto.

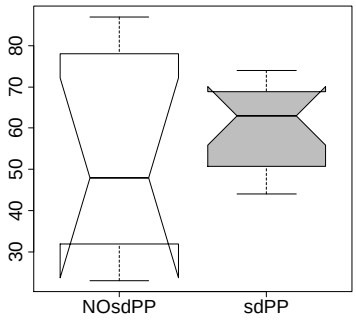


Figura 57 Diagrama de cajas de la variable "Esfuerzo total"

Se ha realizado un t-test de una cola para comprobar si un proyecto de desarrollo tarda más en desarrollarse usando o no sdPP como guía. El resultado nos confirma que no hay diferencias significativas en el tiempo total de desarrollo utilizando sdPP, t(9,381) 0,95, $p > 0.05$. Debido a que tenemos una muestra pequeña (16 grupos que usaron sdPP y 9 que no lo usaron), las diferencias entre las muestras no deberían ser apreciables debido a las limitaciones del t-test.

Se puede afirmar que el uso de sdPP no se incrementa el esfuerzo requerido para desarrollar un proyecto de software, pero sin embargo, con el uso de sdPP se consiguen dos beneficios principalmente: mejor predicción y control en el tiempo de desarrollo, y como hemos visto en la sección anterior, una mejora sustancial en la calidad del producto final.

Cabe señalar que el tiempo total es un agregado del esfuerzo dedicado a entrenar en el uso del sdpFramework y en actividades de desarrollo. Durante la validación del experimento, fue la primera vez que los participantes desarrollaban un proyecto aplicando el conocimiento que esta encapsulado en un sdPP, es de suponer que en futuros desarrollos no se requerirá este tiempo de entrenamiento en sdPP, por lo que se espera que la curva de aprendizaje se termine aplanando y por lo tanto se conseguirá unos desarrollos en menos tiempo.

4.3.4 Utilidad de los elementos de conocimiento propuestos por el modelo sdPP

Los elementos de conocimiento propuestos por el modelo sdPP van a ser estudiados desde dos puntos de vista distintos: desde el punto de vista únicamente del modelado del conocimiento y desde el punto de vista global de las tres fases del ciclo de vida analizadas en la validación de esta tesis doctoral (modelado, adaptación y uso). Para ello en la primera parte de este análisis se centrará en la primera parte de la experimentación (Formalización de conocimiento en sdPPs), mientras que para la segunda parte se estudiará la utilidad de una forma más global, siendo necesario estudiar la utilidad en ambas fases de la experimentación (Formalización y utilización del conocimiento de sdPP)

A. Utilidad de los elementos de conocimiento de sdPP desde el punto de vista del modelado de información

La utilidad de los elementos propuestos por el modelo sdPP han sido evaluados por doce ingenieros de software entrenados en este modelo de conocimiento, teniendo siempre en consideración los siguientes atributos: utilidad, relevancia, satisfacción, facilidad y frustración. La definición de cada uno de estos atributos es:

- **Utilidad**, representa la precepción global de cuánto de provechoso es un sdPP para ayudar a los ingenieros de software a implementar proyectos específicos de software en sus organizaciones.
- **Relevancia**, es la evaluación subjetiva de cómo de apropiado es cada elemento de conocimiento de un sdPP desde el punto de vista de cuándo y cómo usarlos.
- **Satisfacción**, representa la evaluación subjetiva de los ingenieros de software sobre si es capaz de cubrirse las perspectivas puestas en cada uno de los elementos del modelo sdPP.

- **Frustración**, mide el grado de desagrado cuando los participantes tienen que interpretar cada uno de los elementos incluidos en la especificación de los sdPP.
- **Facilidad**, mide la ausencia de problemas cuando los participantes tuvieron que aplicar cada elemento de sdPP.

La evaluación global de los elementos sdPP para ayudar el desarrollo de las actividades del proyecto software fue positivo; 15 (47%) de los patrones tuvieron la máxima puntuación, con una media de 3.92 sobre 5 y para todas las tipos de sdPP.

La Tabla 19 muestra las puntuaciones globales y la media por cada tipo de patrón producido durante la fase de modelado en el estudio empírico. Nota: la puntuación de la frustración esta invertida para que sea comparables con el resto de las variables, ya que el resto de las variables miden cualidades positivas.

Tipo de patrón	sdPP tipo 1	sdPP tipo 2	sdPP tipo 3	sdPP tipo 4	Media
Utilidad	3.78	4.22	3.67	4.00	3.92
Relevancia	3.36	3.42	3.18	3.41	3.34
Satisfacción	3.48	3.68	3.43	3.72	3.58
Facilidad	2.23	2.46	2.37	2.51	2.39
Frustración	2.86	2.94	2.64	3.28	2.93

Tabla 19 Resumen de las variables de utilidad por tipo de patrón

Considerando cada elemento de información propuesto por el modelo de sdPP separadamente, se pueden extraer varias conclusiones: el elemento de información más problemático es claramente los metadatos debido a que necesitan un conocimiento y una experiencia superior para que puedan ser modelados satisfactoriamente. El workflow y el productflow fueron los siguientes elementos más problemáticos en su modelado, aunque fueron bien evaluados por su utilidad ya que contribuyen ofreciendo mucha ayuda a la hora de usar un sdPP.

Para identificar la relevancia de los elementos de conocimiento de un sdPP para ayudar al desarrollo de software, es necesario analizar la relevancia de estos elementos a la hora de identificar cuando es necesario aplicar un patrón de proyecto y cómo adatarlos al contexto del proyecto concreto y sus restricciones. Para conseguir este objetivo se han ajustado varias regresiones lineales sobre los valores de relevancia de los elementos de conocimiento del sdPP.

El primer análisis de regresión sirve para discutir los elementos de patrón de proyecto que son importantes para determinar el patrón más apropiado para un proyecto de software concreto. Las variables que intervienen en la regresión son: cómo usar la descripción (PD), cómo usar el productflow (PF) y el valor en negativo sobre cómo usar los metadatos (M). El coeficiente de regresión entre las variables es alto (R^2=0.762). La ecuación que representa esta regresión es:

Cuándo aplicar un sdPP = -1.6322 + 0.91981 PD + 0.62539 PF – 0.16265 M

a) El primer predictor para saber cuándo usar un sdPP es: cómo usar la descripción. El elemento descripción explica a tipo de problemas se le puede aplicar la parte de la solución del patrón. La percepción de la utilidad del patrón es mayor si el sujeto sabe como usar el elemento de descripción.

b) El siguiente predictor, cómo usar el elemento productflow, el cual esta en la parte de la definición del patrón. La utilidad del sdPP depende directamente con el productflow que describe el flujo de los productos entre las actividades dentro del workflow. Por lo tanto, se puede decir que esta variable también está relacionada con cómo usar el workflow.

c) El último predictor, cómo usar los metadatos, está también en el lado de la definición de la solución del patrón. En este caso, el predictor es negativo. Los participantes en el estudio empírico trabajaron muy duro para intentar modelar los metadatos y no consiguieron entender su utilidad, por lo que su presencia se consideró como un inhibidor de cuando saber usar un sdPP. La percepción negativa de la información de los metadatos durante el estudio se debió a varios factores: esta información es muy difícil de determinar porque requiere un personal altamente experimentado en gestión de proyectos software; los participantes consideraron que la información representada en los metadatos podría modelarse como requisitos, riesgos o "to-dos"; ninguno de ellos tuvieron una idea clara sobre los valores que asignar a los metadatos.

Se ha ajustado una segunda regresión, la cual es relevante para discutir los elementos de sdPP que son importantes para determinar cómo adaptar los patrones a un proyecto específico de software. Estos elementos son workflow (WF), productflow (PF) y "to-dos" (TD). El coeficiente de regresión entre las variables es relativamente alto (R^2=0.674). La ecuación que representa esta regresión múltiple es:

Cómo aplicar un sdPP = -1.8516 + 0.5381 PF + 0.4858 TD + 0.4437 WF

a) El elemento productflow guía al usuario en sobre qué productos son las entradas y las salidas de cada actividad. La percepción de cómo aplicar un sdPP es mayor si el ingeniero de software pueden establecer una correspondencia clara entre las restricciones del proyecto y las actividades a implementar para crear los productos creados por el sdPP.

b) El segundo predictor que ayuda a determinar si un sdPP ayuda a decidir cuando debería aplicarse es la información de los "to-dos". Cómo este tipo de información proporciona lecciones aprendidas en como resolver las restricciones para implementar un sdPP en un proyecto, los ingenieros de software son capaces de determinar fácilmente si un proyecto tiene las mismas restricciones; y si es así seleccionarán el sdPP concreto.

c) Finalmente, la información del workflow es percibida como un elemento que ayuda saber cuándo aplicar un sdPP porque los ingenieros de software pueden interpretar fácilmente si el workflow propuesto por el sdPP es apropiado para completar el calendario de proyectos y planificar las restricciones especificas del proyecto de software.

B. Utilidad de los elementos de conocimiento de sdPP desde el punto de vista global de las fases del ciclo de vida analizadas.

Esta sección presenta un breve análisis de la utilidad de cada elemento del modelo sdPP durante su ciclo de vida. Las etapas que se han considerado para la evaluación de la utilidad han sido: modelado, adaptación y aplicación (o uso).

	Fase de modelado	Fase de utilización	
		Paso de adaptación	Paso de uso
Descripción	7.64	1.8	2.86
Workflow	8.14	6.43	7.14
Productflow	6.57	3.21	4.64
To-dos	6.86	6.79	5.71
Requisitos	4.93	2.86	2.5
Riesgos	4.57	0	0.36
Metadatos	3.07	0	0

Tabla 20 Evaluación de la utilidad de los elementos de conocimiento incluidos en el modelo sdPP

La Tabla 20 (Los valores por encima de 5 se han remarcado en gris), muestra la media percibida sobre la utilidad de cada uno de los elementos de sdPP y por fase. A fin de evaluar si estos valores de utilidad son consistentes entre ellos y pueden ser usados para realizar interpretaciones, se ha calculado la correlación entre los valores de las variables de utilidad entre las tres etapas. Las correlaciones obtenidas se presentan en la Tabla 21

Fase de utilización		
Paso de adaptación	Paso de uso	
0.70 (p<0.1)	0.89 (p<0.05)	Fase de modelado
-	0.92 (p<0.01)	Paso de adaptación (Fase de utilización)

Tabla 21 Correlaciones entre las variables que miden la utilidad en las tres fases analizadas

Los valores de la Tabla 21 muestran que la utilidad percibida por los elementos de sdPP es similar independientemente de la fase en la que se usó. Este resultado se obtuvo comprando las distribuciones de la utilidad de los elementos de conocimiento que componen el sdPP a través de las distintas fases de la aplicación del sdpFramework. Dos distribuciones se pueden considerar similares cuando existe una correlación entre ellas. Las distribuciones de utilidad son algo similares entre las fases de modelado y adaptación ($\rho = 0{,}70$) y entre el entre las fases de modelado y aplicación ($\rho = 0{,}89$). Pero existe una correlación muy alta entre las fases de adaptación y aplicación, ya que los participantes que proporcionaron las puntuaciones fueron los mismos en estas dos fases, por lo que se esperaba que fuesen consistentes ofreciendo una correlación muy alta ($\rho = 0{,}92$). Los p-valores indican que esas correlaciones son significantes.

Cabe señalar que la media de la puntuación de la utilidad está cerca de ser el doble de grande en la fase de modelado. Gran parte de esta diferencia puede explicarse con el valor casi nulo que se ha dado a los elementos de información "riesgos" y "metadatos" en las dos últimas fases, lo que podría atribuirse al hecho de que los sujetos se les asignó un sdPP en particular, y no darles la opción de elegir uno u otro. Por lo tanto, los metadatos no tienen una utilidad clara para ellos cuando se escoge un sdPP, y los riesgos no podrían ser usados ya que se les asignó un sdPP en concreto.

Las razones para esta diferencia en la evaluación de la utilidad en las fases de modelado y las otras fases del ciclo de vida del sdPP fueron:

- Los autores de los sdPP (participantes en la fase de modelado) tuvieron un mejor entendimiento y una opinión sobre el valor añadido aportado por cada activo incluido en el modelo de datos de sdPP.

- Los participantes en la fase de uso se evaluó la utilidad de la información incluida en los sdPPs obtenidos de la fase de modelado, así como el valor potencial de los activos de conocimiento incluidos en el modelo sdPP, para guiar en las actividades de desarrollo de software.

La Figura 58 muestra la similitud en las distribuciones de la utilidad a través de las diferentes evaluaciones: fase de modelado y uso (incluyendo los pasos de adaptación y la aplicación).

Se ha observado una gran diferencia en la percepción de la utilidad entre las fases de modelado y uso, así como los elementos de información "requisitos" y "riesgos". Como hemos visto en el capítulo 3 (Solución propuesta) en el apartado 3.2 (Modelo de conocimiento sdPP), un patrón es una composición de la definición de un problema y la descripción de la solución; durante la fase de modelado todos los elementos eran igualmente importantes debido a que el propósito de esta fase es proporcionar un correcto modelado en todos los elementos de conocimiento, independientemente si los elementos de conocimiento pertenecían a la solución o al problema. Sin embargo, durante los pasos de adaptación y aplicación en la fase de uso, los participantes tenían que elegir el patrón que mejor se ajustaba para desarrollar la aplicación. Pero solo había una opción disponible por cada tipo de proyecto, por lo que el proceso de selección fue muy simple. En consecuencia, los elementos que describen el problema (descripción, requisitos y riesgos) no se evaluaron como muy útiles. Los elementos de conocimiento "requisitos" y "riesgos" serían mejor valorados si hubiera un número mayor de patrones donde poder elegir.

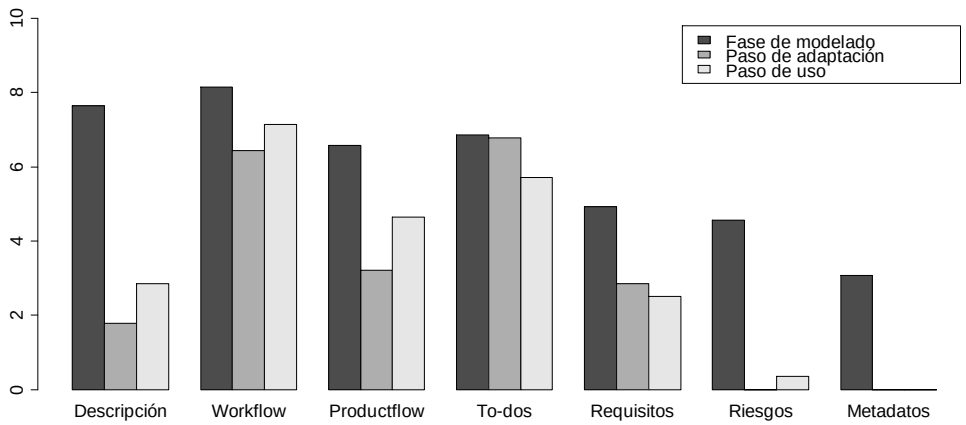


Figura 58 Utilidad de los elementos de sdPP por fases

Sin embargo, los elementos que describen la solución (workflow, productflow y "to-dos") fueron muy bien valorados en las tres fases debido a que estos elementos de conocimiento proporcionan mucha información sobre el desarrollo de software durante las tres fases analizadas.

El productflow se calificó con valores más bajos que el resto de los elementos que describen la solución. El productflow proporciona más información que el workflow, pero muchas veces los participantes percibieron esta información como valores redundantes y recibieron valores más bajos.

4.3.5 Validez de los resultados

En esta sección se analizará la validez de los resultados del caso de estudio embebido ejecutado en esta tesis doctoral según se describen en el trabajo de (Juristo & Moreno, 2001) y (Yin, 2009). Esta validez de los resultados se describirá por las sub-unidades de análisis que conforman el caso de estudio embebido mostrados en Figura 49 y Figura 50.

a) Primero, se analizarán las amenazas relacionadas con la precisión de la definición del estudio empírico para discutir las preguntas de investigación.

   Una posible amenaza a la validez sería el sesgo introducido por los dos expertos cuando realizaron los juicios de la corrección de los elementos de conocimiento de sdPP de cada uno de los modelos realizados por los participantes. Se ha realizado un test de Kappa de Cohen entre las evaluaciones de los dos expertos, el resultado obtenido fue un "alto consenso" (κ=0.6); por lo que no se aprecia ningún tipo de sesgo introducido por los expertos. A pesar de que aparentemente hay una gran variabilidad entre las evaluaciones de los expertos, se ha realizado un análisis de varianza para comprobar si hay diferencias significativas entre los distintos tipos de sdPP. El resultado fue negativo (ANOVA, $F(3, 32)=0.327$, $p=0.806$), pudiéndose determinar que no existe una variabilidad estadísticamente significativa entre los dos juicos de los expertos.

   Otra posible fuente de sesgo podría ser las fuentes bibliográficas elegidas por los participantes para modelar los sdPP. Se ha realizado un análisis de las fuentes bibliográficas y existe una variabilidad bastante alta entre ellas. Por otra, los participantes usaron una media de siete fuentes bibliográficas, donde casi el 75% de las referencias se encuentran al menos en otro conjunto bibliográfico de otro participante, y el otro 25% de las referencias son totalmente diferentes. Por lo tanto, cualquier

diferencia en la importancia de la bibliográfica utilizada puede ser atribuida a la elección de los participantes.

La información cualitativa obtenida durante el estudio empírico ha sido útil para analizar la evolución de la actitud y aptitud de los participantes que modelaron los sdPPs. Sin embargo, la información cuantitativa adicional podría ser útil para obtener cifras más precisas sobre los factores humanos que influyen en el modelado de los sdPPs, tales como: la cultura de compartición de conocimiento, capacidades de trabajo en equipo, capacidades de abstracción y capacidades para formalizar los ítems de conocimiento.

En el caso empírico se modelaron cuatro tipos distintos de prácticas efectivas de desarrollo software, desde pequeños y medianos, metodologías para desarrollo de aplicaciones móviles, hasta metodologías para desarrollo de grandes sistemas de información de negocios, incluyendo también prácticas para la adaptación e integración de software ya existente. Aunque, la propuesta sdPP puede ser usada para modelar patrones de otros tipos de metodologías, buenas prácticas, marcos de referencia, etc. Se podrían conseguir conclusiones mucho más refinadas y más relevantes si se hace un estudio más amplio con más tipos de sdPPs.

La validación experimental se centró en analizar los efectos de la utilización de sdpFramework en el esfuerzo requerido para desarrollar un proyecto de software y las diferencias en la calidad de los productos producidos durante el proyecto.

En primer lugar, una posible amenaza a la validad puede ser la tendencias de los dos expertos para evaluar la calidad de los proyectos desarrollados. El cumplimiento de los requisitos se evaluó a través de un cuestionario del tipo si/no por lo que fue muy sencilla su evaluación sin posibilidad de introducir sesgo.

En segundo lugar, las cifras sobre el esfuerzo empleado en el desarrollo de las actividades del proyecto fuero ofrecidas por los propios participantes. Para evitar cualquier sesgo en relación a estas cifras, los expertos hicieron un seguimiento de la información ofrecida sobre el esfuerzo durante la ejecución del experimento para asegurarse que las evaluaciones de esfuerzo concuerdan con la carga de trabajo de cada uno de los proyectos de desarrollo.

b) En segundo lugar, se ha realizado un estudio de las amenazas relacionadas con el grado en que las afirmaciones causa-efecto son válidas.

Existen varias limitaciones en cuento al número de participantes, número de patrones y tipo de patrones. A fin de abordar estos problemas, el estudio empírico sigue un diseño ortogonal y fraccional, donde cada uno de los doce participantes formalizó tres tipos de sdPPs de los cuatro en total, tal y como se describe en la Tabla 9.

La utilidad de la propuesta de sdPP para gestionar adecuadamente el conocimiento relacionado con las prácticas efectivas de desarrollo de software también se evaluó a través la percepción subjetiva de los participantes que participaron en el estudio empírico, donde en ningún caso la opinión de los investigadores fue impuesta a los participantes, si no que los participantes expresaron su opinión durante el modelado y la evaluación de la utilidad de los sdPPs. Los expertos que ejecutaron el estudio empírico compararon la distribución de la percepción de la utilidad entre los participantes que pedían ayuda y los que no lo hicieron y no se encontraron diferencias significativas por lo tanto se puede concluir que los expertos no introdujeron ningún tipo de sesgo en las opiniones de los participantes.

Para identificar la mejora de la calidad y la ausencia de costos a lo largo de la aplicación del sdpFramework, se diseño un experimento donde los datos de un grupo de control compuesto por 18 participantes fueron utilizados para la validación experimental. Los datos pueden ser considerarse suficientes para identificar las cuestiones relevantes que pueden incluir en el modelado y el uso de patrones de proyecto cuando se aplica sdpFramework. De esta manera, los 50 participantes en la segunda fase (18 personas en el grupo de control y 32 personas en el grupo de test) es representativa para validar el fenómeno estudiado.

c) En tercer lugar, se analizaron las amenazas que proceden de las generalizaciones que se pueden realizar de los resultados obtenidos.

Existen varias limitaciones contextuales referentes a la experiencia previa de los participantes. Todos los participantes son ingenieros junior de software, por lo que son una muestra representativa de especialistas en procesos de desarrollo software. Aunque cabe esperar mejores resultados en la calidad de los sdPP modelados si los sujetos tienen mayor experiencia en ingeniería de procesos de desarrollo.

4.3.6 Discusión

Se han estudiado muchos trabajos de investigación en el estado del arte relacionados con los patrones de proceso que discuten las mejoras en la calidad del software producidas por su uso (Aurum, 2008). Sin embargo, es importante

remarcar que la efectividad de los patrones depende de la correcta adaptación al contexto especifico de cada una de las organizaciones de desarrollo de software (Medina-Dominguez et al., 2010). En varios casos, la aplicación de los patrones de procesos está basada en la adopción de patrones creados por expertos ajenos a la organización. En comparación con otros, este trabajo de investigación está centrado en la creación de nuevos patrones de proceso por la plantilla de una organización de desarrollo software integrando conocimiento prexistente de su propia experiencia tácita.

Las implicaciones de estos resultados pueden ser analizados desde dos perspectivas: operacional y organizativa. La primera perspectiva esta centrada en la discusión como facilitadores para implementar actividades especificas para el modelado y el uso del conocimiento de los sdPPs. La segunda perspectiva está centrada en discutir las implicaciones organizativas requeridas para implementar efectivamente una iniciativa parecida al sdPP.

a) **Perspectiva operacional**

Los resultados obtenidos durante el estudio empírico indican que el proceso de formalización de sdPPs tiene que centrarse en facilitar la adopción de más prácticas efectivas para un proyecto de software concreto. Para poder conseguir este objetivo, el conocimiento encapsulado en un sdPP debe estar preparado especialmente para:

i. Buscar el patrón más apropiado para un proyecto específico de desarrollo software. Los participantes en iniciativas basadas en patrones de procesos deben prestar una atención especial a configurar el elemento descripción e incluir palabras clave que identifiquen los diferentes tipos de proyectos que pueden gestionar con el patrón concreto. Esta simple práctica contribuye a incrementar la efectividad de la aplicación de variadas formas de búsqueda soportadas a través de la aplicación la tecnología ofrecida por sdpFramework.

   Los participantes del estudio empírico además consideraron el productflow como un elemento relevante de conocimiento para encontrar el sdPP más apropiado para un proyecto concreto. Para ser efectivos, el productflow debe de contener información clara de los productos y de las dependencias entre ellos. Esta práctica contribuye a mejorar el uso de las funcionalidades de recuperación tales como la búsqueda gráficas y búsquedas por ejemplo. Como indican los resultados sobre la utilidad de los sdPP, estos mecanismos de recuperación son esenciales para

determinar el patrón más apropiado para cada proyecto de desarrollo software.

En los resultados también incluyen una percepción negativa de la utilidad de los metadatos para encontrar el patrón más recomendable para un proyecto concreto de desarrollo software. Estos elementos proporcionan un valor numérico de las restricciones contextuales asociadas al proyecto software como propuestas paramétricas de métodos de estimación de proyectos software. Los ingenieros de software involucrados en el estudio empírico no tuvieron suficiente entrenamiento en catalogar proyectos de software desde este punto de vista, por lo que no fueron capaces de entender como asignar e interpretar valores de esta categoría. De esta manera, para promover el uso de los metadatos para facilitar las búsquedas de sdPPs, es necesario adaptar la categoría de metadatos al contexto organizativo y proveer ingenieros de software con un mayor conocimiento y entrenamiento en las categorías propuestas por los metadatos.

ii. Adopción y aprendizaje de prácticas efectivas de ingeniería de software. Uno de los principales objetivos de los patrones de procesos es facilitar la adopción y aprendizaje de las practicas requeridas para llevar a cabo proyectos software desarrollados por una organización de ingeniería software. En la mayoría de los casos, estas prácticas son obtenidas de conocimiento prexistente que incluye información de actividades (incluyendo workflows) de ingeniería del software, productos (incluyendo productflows) y lecciones aprendidas. Como indican los resultados, la corrección de estos tipos de conocimiento está influenciado para la relevancia de las referencias prexistentes consideradas para la formalización de un sdPP.

   Para facilitar la inclusión de estas prácticas efectivas desde el estado del arte, es necesario utilizar un mecanismos para diseminar, de una forma formalizada, de practicas efectivas, tales como las que propone (Sanchez-Segura et al., 2010).

   Además para facilitar el intercambio de prácticas efectivas entre las unidades de desarrollo software de la misma compañía o grupo de compañías, es necesario promover la implementación de mecanismos para el intercambio de conocimiento. En el ámbito de este trabajo de investigación, se ha implementado SPEM (OMG, 2008), pero ha tenido que ser adaptado porque su especificación actual no permite incluir fácilmente elementos tales como "to-

dos", requisitos, y riesgos que fueron evaluados positivamente por los participantes del estudio empírico. Esto constituye un problema que debe ser resuelto para facilitar el intercambio de buenas prácticas entre los ingenieros de software.

iii. Adaptación de las prácticas a las restricciones de un proyecto concreto de desarrollo software. Estos elementos incluidos en un sdPP deberían proporcionar a los ingenieros de software conocimiento en como adaptar las prácticas efectivas de desarrollo software a las restricciones de un proyecto concreto. Tal y como describe la propuesta sdPP, este tipo de conocimiento esta incluido en los requisitos, riesgos y "to-dos". Sin embargo, es necesario tener en cuenta que según el análisis de corrección de un sdPP obtenido durante el estudio empírico indica que estos elementos tienes un nivel medio o bajo de calidad, esto es debido a que la creación efectiva de estos elementos de conocimiento requieren la combinación de una amplia experiencia en la aplicación de prácticas efectivas en el área de gestión de proyectos, habilidades específicas y capacidades para presentar este conocimiento de un modo accesible a los ingenieros de software que necesiten este tipo de conocimiento tácito.

   Los participantes de este estudio empírico consideraron muy importante el conocimiento tácito obtenido en los cursos de ingeniería de software en la universidad, ya que el estudio de estas metodologías generaban discusiones que servían para poder aprender ese tipo de conocimiento de una forma más práctica de dichas metodologías.

   Una definición más refinada de los elementos de información tales como ejemplos, discusiones y lecciones aprendidas (García, 2011) contribuiría a una elicitación más efectiva del conocimiento tácito de los ingenieros de software.

   Sin embargo, la elicitación de conocimiento tácito necesita un esfuerzo mayor de los ingenieros de software, por lo que esto es una barrera que hay que prevenir para la aplicación de propuestas parecidas a sdPP.

b) **Perspectiva organizativa**

Como resultado de una evaluación final de los mecanismos de los patrones de proyecto, los participantes y los expertos coincidieron que el un factor crítico para el éxito para conseguir patrones de proyectos correctos fue la aplicación efectiva de patrones colaborativos entre los

autores de patrones de proyecto y los expertos. Los patrones de colaboración considerados como útiles fueron los que tuvieron que ver con la comunicación síncrona/asíncrona así como compartir el conocimiento. Estos tipos de estrategias de colaboración facilitan a los diseñadores de patrones de proyecto a contribuir a generar comentarios en foros de discusión para poder mejorar el aprendizaje en el modelado de patrones de proyectos.

Basados en estas discusiones, cada autor de un patrón de proyecto pudo formalizar mejor el conocimiento tácito y explícito a incluir en cada patrón de proyecto.

Los participantes del estudio empírico implementaron estos tipos de discusiones y fueron evaluados positivamente porque contribuyeron a compartir conocimiento sobre como modelar patrones de proyecto que permitían difundir conocimiento táctico en el modelado de patrones. Además, los participantes que hicieron más hincapié en compartir conocimiento fueron los que proporcionaron patrones con un nivel de corrección superior.

La creación de conocimiento durante la fase de elicitación de patrones ayudó a implementar una estrategia de gestión de conocimiento promovida a través de la aplicación de la solución sdPP, enriqueciendo el capital intelectual de las organizaciones de software.

Esta evolución empieza desde el conocimiento individual de las prácticas efectivas de la ingeniería del software hasta el conocimiento adaptativo de las organizaciones que facilitan el intercambio de conocimiento entre los diferentes equipos de la organización de una manera sostenible.

Los mecanismos para compartir conocimientos inherentes en las fases del ciclo de vida del sdPP también permiten la transferencia sostenible de conocimiento desde ingenieros de software experimentados a los ingenieros juniors. De esta manera, la organización de ingeniería de software está más preparada para adaptar sus procesos internos a los cambios tecnológicos que afectan a su competitividad (Mora-Soto et al., 2010).

Estos descubrimientos sugieren la necesidad de crear modelos adicionales para facilitar la transferencia y reuso del conocimiento adjunto en un sdPP. Estos modelos deberían incluir mecanismos para alinear iniciativas parecidas a sdPP con los objetivos operacionales de una organización de ingeniería de software para evaluar la contribución de estas iniciativas al rendimiento organizativo (hechos que no han sido

estudiados en esta tesis doctoral ya que no entraba dentro de su ámbito de investigación)

Para facilitar la implementación de soluciones parecidas a sdPP, se ha descubierto que es fundamental diseminar los objetivos, estrategias, roles, responsabilidades y beneficios; entre todas las partes interesadas. Es muy importante maximizar la visibilidad del proceso para compartir el conocimiento ya que este tipo de propuestas requieren un cambio cultural en los participantes involucrados. Aunque, esto no es un proceso complejo, requiere de unos cambios internos en las actitudes de los participantes, lo cual no es fácil de conseguir.

Finalmente, es necesario indicar que la aplicación de una iniciativa del estilo de sdPP requiere una inversión adicional en herramientas de gestión del conocimiento, pero como indica (Sanchez-Segura et al., 2011), el rendimiento de equipos de desarrollo software mejoran el nivel de calidad si no se hacen planificaciones largas, optimizando el valor añadido que se ofrece a los clientes de las organizaciones de desarrollo software.

CONCLUSIONES Y TRABAJOS FUTUROS

Tabla de contenidos: Conclusiones

5: CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se van a presentar las conclusiones a las que se pueden llegar tras el trabajo de investigación realizado en la presente tesis doctoral. Las conclusiones están organizadas según las preguntas de investigación presentadas en el capítulo 4 Validación.

A continuación se presentará las posibles propuestas de trabajos futuros que han surgido tras realizar el trabajo de investigación presentado en esta tesis doctoral.

5.1 Conclusiones

En la literatura estudiada podemos encontrar trabajos de investigación que analizan el uso de patrones de proceso como elementos para mejorar la calidad de los productos de desarrollo software (Aurum, 2008), pero no se ha encontrado estudios que analicen la utilidad de patrones de procesos de desarrollo software orientados a solucionar el desarrollo completo de un proyecto software. Además en la literatura estudiada no se ha encontrado referencias donde se hagan un estudio de cuales son los factores que ayuden al modelado y creación de información. Los resultados en esta tesis doctoral indican que el correcto uso de patrones de proyecto en organizaciones de desarrollo software dependen en la correcta adaptación de las condiciones contextuales de la organización y a las características específicas de los proyectos de software desarrollados. La originalidad de este trabajo de investigación está enfocada en la creación de nuevos patrones de proyecto, creados por los empleados de una organización de desarrollo software, mediante la integración de los conocimientos pre-existentes y su propia experiencia tácita.

El framework presentado en esta tesis doctoral constituye un nuevo enfoque para la creación y gestión de conocimiento para posibilitar a los ingenieros de software la adopción de prácticas efectivas para el desarrollo de proyectos de software. Un sdPP incluye elementos de conocimientos que describen cómo desarrollar un proyecto de software (en la parte de la solución del patrón con elementos como, workflow, productflow y to-dos y riesgos); los proyectos para los que es válido el uso de dicho patrón (en la parte de la descripción del problema del patrón, con elementos como: descripción, metadatos y requisitos)

En las conclusiones analizaremos las preguntas de investigación, tal y como se describieron en el capítulo 4:

- **Pregunta de investigación 1:** ¿Cuáles son los factores que influencian el correcto modelado de sdPPs?
- **Pregunta de investigación 2:** ¿La aplicación del conocimiento de un sdPP aumenta la calidad de los productos de desarrollos software?
- **Pregunta de investigación 3:** ¿La aplicación de las actividades para aplicar el conocimiento de un sdPP aumenta el esfuerzo en el desarrollo de un proyecto software?
- **Pregunta de investigación 4:** ¿Cuál es la utilidad de los elementos de información proporcionados por sdPP a través de las fases del ciclo de vida de los patrones?

5.1.1 Factores que afectan al correcto modelado de sdPPs

Según el análisis estadístico llevado a cabo en la experimentación de esta tesis, los factores que influyen en el correcto modelado de sdPP dependen de varios factores.

El primero de ellos consiste en tener en las fuentes bibliográficas una buena definición del productflow y del workflow, así como la inclusión de "to-dos" que proporcionen información con valor añadido relacionado con la experiencia previa en la implementación de las prácticas efectivas de desarrollo software en proyectos de desarrollo similares.

La exactitud de las definiciones del productflow y del workflow está claramente influenciada por la relevancia de las referencias bibliográficas usadas para ayudar en las actividades de formalización de un sdPP. La relevancia de las referencias bibliográficas depende del grado de formalidad en presentar las actividades requeridas para implementar las prácticas, los productos que se deben generar en cada una de las actividades y las relaciones entre las actividades y los productos. En este sentido, las mejores referencias para tener en consideración durante la formalización de sdPP son las definiciones previas de los procesos desplegados en una organización y las referencias que cumplen los criterios de exactitud mostrados en el capítulo 4 validación.

La corrección de un sdPP también depende del grado de colaboración y de discusión entre los autores de los patrones. Este tipo de colaboración es esencial para resolver dos problemas colaborativos que aparecen cuando se define un sdPP:

a) La identificación de la información que es realmente necesaria y útil para los ingenieros de software cuando implementan un tipo específico de proyectos.

b) La identificación del camino más apropiado para formalizar cada elemento de información en un sdPP y los elementos incluidos en el modelo sdPP más apropiados para registrar cada elemento de conocimiento.

Finalmente, la corrección de cada patrón modelado también depende de la experiencia previa del personal a cargo de la creación de un sdPP. En esta área, los factores más importantes están relacionados con los años de experiencia en el área de la organización, el conocimiento práctico de los "todos" incluidos en un patrón y entrenamiento en las prácticas sobre métodos de desarrollo de software.

5.1.2 Mejora de la calidad en los productos desarrollados con sdPP.

Existen varios trabajos de investigación que indican que el uso de patrones de procesos son un precursor de la calidad en proyectos de desarrollo software (Medina-Dominguez, 2010). El análisis estadístico de los resultados obtenidos durante la validación de esta tesis doctoral proporcionan las mismas conclusiones.

Los efectos específicos proporcionados por el uso de sdpFramework identificados en este trabajo tienen que ver con la mejora de la calidad de los productos intermedios entre las actividades del workflow; tales como la especificación de requisitos, documentos de diseño y análisis, diagramas de arquitectura y componentes, etc. Esta mejora se debió a que los modelos de sdPPs utilizados para el desarrollo de software incluyen información concreta y clara sobre las actividades, productos, workflows y productflows. Esto es debido a que el modelo de sdPP contiene elementos de información efectivos, eficientes y fáciles de utilizar por los ingenieros de software; tal y como hemos visto en el capítulo 4.

5.1.3 Análisis del esfuerzo empleado en aplicar las actividades para aplicar el conocimiento de un sdPP en un proyecto de desarrollo software.

Las evidencias obtenidas de la validación indican que la implementación y uso de sdpFramework para la gestión del desarrollo de proyectos de software requiere de un esfuerzo para educar en los principios, conceptos y el modelo de sdPP, así como la herramienta sdpReuser. No obstante, según las evidencias estadísticas descritas en el capítulo 4 "Validación" la diferencia entre los grupos de control y de test no son estadísticamente significativas. Por lo que se puede

concluir que la aplicación del entorno sdpFramework no va a tener unos sobrecostes significativos con respecto a no utilizar un sistema de gestión de conocimiento.

En el caso de los proyectos que no usaron el entorno sdpFramework, fue necesario emplear mucho más esfuerzo para buscar prácticas eficientes de desarrollo que fuero útiles para el desarrollo del proyecto y para aprender como adaptarlas a un proyecto concreto. Este esfuerzo fue menor que el requerido para aprender el entorno sdpFramework, pero las prácticas no fueron de tanta calidad como las que ofrece un sdPP.

Para concluir, el esfuerzo empleado en llevar a cabo las actividades del proyecto fueron muy similares. Por lo tanto y tras un estudio estadístico pertinente, se puede afirmar que el uso del entorno sdpFramework no necesita de un esfuerzo adicional en actividades específicas de ingeniería del software para ejecutar un proyecto de desarrollo.

Además, los resultados obtenidos muestran que el uso del entorno sdpFramework permitiría realizar estimaciones de esfuerzo y de tiempo mucho más previsibles que no utilizando ningún sistema de gestión de conocimiento. Las evidencias estadísticas que apoyan esta afirmación nos indica que los equipos que utilizan sdPP, la desviación típica es significativamente menor que la desviación típica de los grupos que no utilizan sdPP. Una desviación típica inferior, en este caso, indica que el esfuerzo es la estimación de esfuerzo es más fiable.

5.1.4 Utilidad de los elementos de conocimiento propuestos por el modelo sdPP

Según las preguntas de investigación sobre la utilidad de los elementos de conocimiento en un sdPP, se puede concluir que depende de la facilidad de identificar cuándo y cómo aplicar un patrón específico en un contexto de un proyecto de desarrollo software.

Varios elementos de conocimiento como la descripción y el productflow están considerados como útiles para identificar cuando adaptar un patrón de proyecto específico. La elaboración de estos elementos debería estar orientada a facilitar la aplicación correcta de distintas técnicas de búsqueda para favorecer la aplicación correcta de sdPPs. El elemento de información “descripción” debería contener un conjunto de palabras clave que identifique los diferentes tipos de proyectos de desarrollo para los que el patrón es válido. El productflow debería estar claramente definido, identificando la secuencia de los productos intermedios y finales para desarrollar durante el proyecto, permitiendo la aplicación efectiva de la búsqueda gráfica de este tipo de elementos. Se puede

concluir que los metadatos no son útiles a menos que se adapten para diferenciar las restricciones especificas de proyectos similares que gestiones una misma organización de software para que los ingenieros de software tengan competencias especificas para asignar e interpretar sus valores correctamente.

Los elementos que más aportan información para saber cuándo y cómo adaptar un patrón a las restricciones específicas de un proyecto de software son los workflows, productflows y "to-dos". La utilidad de estos tipos de elementos se debe a dos factores:

a) Los workflows y los productflows facilitan el aprendizaje y la adopción de nuevas prácticas.

b) Las lecciones aprendidas contribuyen a saber cómo resolver problemas específicos para aplicar las prácticas efectivas a las restricciones del desarrollo.

5.2 Trabajos futuros

En esta sección se presentarán varias propuestas de trabajos futuros que han surgido tras la ejecución del trabajo de investigación realizado en la presente tesis doctoral, que por distintas razones no se han podido acometer en esta tesis doctoral. Estas razones son desde que se tiene que acotar un límite en el trabajo desarrollado en una tesis doctoral, nuevas ideas que han surgido durante la ejecución de la tesis, mejoras que se pueden realizar en la investigación, etc. Las propuestas de trabajos futuros que ofrecemos en esta tesis doctoral son:

Realizar más tipos de sdPPs: La primera parte de la experimentación de esta tesis doctoral consistió en el modelado de cuatro tipos de sdPPs. Aunque el conjunto es suficientemente amplio para realizar la experimentación y se han obtenido resultados relevantes, podría ser muy interesante buscar más tipos de metodologías, mejores prácticas, marcos de referencias y conocimiento tácito de empresas o ingenieros de software; para poder hacer más consistente la investigación y tener más tipos de sdPPs para poder aplicar al desarrollo de un proyecto software. Además con más tipos de sdPPs los resultados de las búsquedas en el repositorio de patrones serán más amplios, por lo que se podrán realizar investigaciones referentes a las búsquedas y los elementos que más ayudan a los ingenieros de software para realizar búsquedas sobre el repositorio de conocimiento.

Modelado de patrones por distintos tipos de personas según su cualificación: En el primera parte de la validación de esta tesis doctoral se han modelado distintos tipos de sdPPs por un conjunto de participantes. Este

conjunto de participantes, como demuestran una encuesta demográfica, tienen las mismas características en cuanto a conocimientos en ingeniería informática. El conjunto de los sujetos de la experimentación proceden del mismo curso de la Universidad Carlos III; además los conocimientos sobre procesos de desarrollo software son muy similares. Sería muy interesante realizar una experimentación probando los entre distintos tipos de perfiles dependiendo de sus habilidades con la ingeniería de procesos y estudiar la percepción de los elementos de sdPP

Realizar un análisis con un número mayor de proyectos de desarrollo: En la segunda parte de la validación experimental de la presente tesis doctoral se aplicó el conocimiento de un sdPP a dos proyectos de desarrollo software muy parecidos entre si. Sería interesante realizar un análisis más en profundidad aplicando el conocimiento de sdPPs a un conjunto de proyectos de desarrollo software y de características diferentes. De esta manera se podrá analizar si la propuesta sdPP es capaz de mejorar la calidad sin introducir sobrecostes en el desarrollo de distintos tipos de proyectos de desarrollo de software. Pudiéndose analizar los elementos de conocimiento de sdPP más útiles por tipo de proyecto.

Analizar el uso de sdpFramework en sucesivos desarrollos para allanar la curva de aprendizaje de entorno sdpFramework y poder comprobar que se consiguen mejoras en el esfuerzo. En la experimentación realizada en esta tesis doctoral, tanto en la primera parte (modelado) o en la segunda parte (utilización) se ha utilizado sdpFramework. Para los participantes los experimentos fue la primera vez que trabajaban con este framework, el tiempo empleado en aprender el funcionamiento del modelo sdPP o la herramienta sdpReuser fue contabilizado como esfuerzo y comparado con el grupo de control que no tuvieron que emplear tiempo para aprender ningún framework. Es lógico pensar que si los participantes tuvieran que emplear sdpFramework en proyectos sucesivos, cada vez necesitarían menos tiempo para aprender a usar el framework por lo tanto el esfuerzo podría reducirse con el uso sucesivo de sdpFramework.

Analizar el uso de sdpFramework con respecto a otras técnicas de gestión del conocimiento: En la segunda parte de la validación experimental de esta tesis se utilizó el conocimiento de los sdPPs para acometer un proyecto de desarrollo software. En dicho experimento existía un grupo de control para poder hacer la comparación, ningún equipo del grupo de control utilizó algún patrón de desarrollo, guías, prácticas, marcos de referencia, etc. (excepto un grupo cuyos resultados no fueron destacables). Sería interesante realizar una experimentación que compare los beneficios del uso de la propuesta sdPP con respecto a otras propuestas que existan en la literatura o que se utilicen en el

desarrollo software tales como metodologías ágiles, marcos de referencia, mejores prácticas, etc.

El framework ofrecido por sdpFramework no está diseñado para soportar otro tipo de patrones de procesos, como por ejemplo los orientados a proceso, actividad o producto; debido a que el núcleo de conocimiento en el que orbita el marco metodológico propuesto es el proyecto de desarrollo software completo.

BIBLIOGRAFÍA

6: BIBLIOGRAFÍA

(Alexander, 1979) ALEXANDER C (1979) *The timeless way of building*. Oxford University Press, USA, ISBN: 0-19-502402-8.

(Alexander et al., 1978) ALEXANDER C, ISHIKAWA S, SILVERSTEIN M, JACOBSON M, FIKSDAHL-KING I, ANGEL S (1978) *A pattern language: Towns, Buildings, Construction*. Oxford University Press, USA, ISBN: 978-0195019193

(Allan et al., 2003) ALLAN J, CALLAN J, COLLINS-THOMPSON K, CROFT B, FENG F, FISHER D, LAFFERTY J, LARKEY L, TRUONG TN, OGILVIE P (2003) *The lemur toolkit for language modeling and information retrieval*, URL:http://lemurproject.org.

(Ambler, 1999) AMBLER SW (1999) *More process patterns: delivering large-scale systems using object technology*. Cambridge Univ Pr , ISBN: 0-521-65262-6

(Ambler, 1998) AMBLER SW (1998) *Process patterns: building large-scale systems using object technology*. Cambridge University Press, ISBN: 0-521-64568-9

(Amescua et al., 2010) AMESCUA A, BERMÓN L, GARCÍA J, SÁNCHEZ-SEGURA MI (2010) Knowledge repository to improve agile development processes learning. *IET Software* 4(6), 434–444, DOI: 10.1049/iet-sen.2010.0067

(Amescua et al., 2005) AMESCUA A, CUEVAS AGUSTÍN G, GARCÍA GUZMÁN J, LLORENS J, MARTÍNEZ P, MARTÍN D (2005) A Pattern-Based Approach to Deploy Process Improvements in Small Settings. Proceedings of the First International Research Workshop for Process Improvement in Small Settings, 2005p 157–165.

(Antunes et al., 2007) ANTUNES B, SECO N, GOMES P (2007) Knowledge management using semantic web technologies: an application in software development. Proceedings of the 4th international conference on Knowledge capturep 187 – 188ACM, DOI: 10.1145/1298406.1298447

(Apache Jakarta, 2005) APACHE JAKARTA (2005) *Apache Lucene a high performance, full-featured text search engine library*, URL: http://jakarta.apache.org/.

(Appleton, 1997) APPLETON B (1997) Patterns for conducting process improvement. Proceedings of PLoP97Citeseer.

(Ardimento et al., 2009) ARDIMENTO P, BALDASSARRE MT, CIMITILE M, VISAGGIO G (2009) Empirical Validation of Knowledge Packages as Facilitators for Knowledge Transfer. *Journal of Information & Knowledge Management (JIKM)* 8(03), 229–240, DOI: 10.1142/S021964920900235X.

(Aurum et al., 2008) AURUM A, DANESHGAR F, WARD J (2008) Investigating Knowledge Management practices in software development organisations – An Australian experience. *Information and Software Technology* 50(6), 511–533, DOI: 10.1016/j.infsof.2007.05.005.

(Basili et al., 2002) BASILI V, CALDIERA G, ROMBACH H "Experience factory," in *Encyclopedia of Software Engineering*, John Wiley & Sons, pp. 469–476, 2002, available at: http://www.cs.umd.edu/~mvz/handouts/fact.pdf , DOI:10.1002/0471028959.sof110

(Basili et al., 2007) BASILI VR, BOMARIUS F, FELDMANN RL (2007) Get Your Experience Factory Ready for the Next Decade--Ten Years after "How to Build and Run One"--. *29th International Conference on Software Engineering (ICSE'07 Companion)*, 167–168, DOI: 10.1109/ICSECOMPANION.2007.41

(Berczuk & Appleton, 2003) BERCZUK SP, APPLETON B (2003) *Software configuration management patterns: effective teamwork, practical integration*. Addison-Wesley Professional, ISBN: 0201741172

(Bermón, 2010) BERMÓN L "Librería de activos para la gestión del conocimiento sobre procesos software: PAL-Wiki," 2010.

(Birkinshaw & Sheehan, 2003) BIRKINSHAW J, SHEEHAN T (2003) Managing the knowledge life cycle. *Engineering Management Review, IEEE* 31(3), 19.

(Borges & Falbo, 2002) BORGES L DA MSB, FALBO R DE A (2002) Managing Software Process Knowledge. *Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business and Applications--CSITea'2002*, 227–232.

(Bots & de Bruijn, 2002) BOTS PWG, BRUIJN H DE (2002) *Effective knowledge management in professional organizations: going by the rules*. IEEE Comput. Soc.

(Buschmann et al., 2007) BUSCHMANN F, HENNEY K, SCHMIDT DC (2007) *Pattern-oriented software architecture: On patterns and pattern languages*. John Wiley & Sons Inc, ISBN: 978-0-471-48648-0

(Campbell, 2004) CAMPBELL J (2004) Interaction in collaborative computer supported diagram development. *Computers in Human Behavior* 20(2), 289–310, DOI: 10.1016/j.chb.2003.10.019

(Carey & Carlson, 2002) CAREY J, CARLSON B (2002) *Framework process patterns: lessons learned developing application frameworks*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, ISBN: 9780201731323

(Chau & Maurer, 2005) CHAU T, MAURER F (2005) A case study of wiki-based experience repository at a medium-sized software company. Proceedings of the 3rd international conference on Knowledge capture 185–186.

(Coplien, 2004) COPLIEN J (2004) Patterns of engineering. *Potentials, IEEE* 23(2), 4–8.

(Cunningham & Leuf, 2001) CUNNINGHAM W, LEUF B (2001) *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional, ISBN: 020171499X

(Dingsøyr, 2000) DINGSØYR T (2000) An evaluation of research on experience factory. Proc. Workshop Learning Software Organizations (PROFES 2000) p 55–66 Springer LNCS.

(Feldmann & Carbon, 2003) FELDMANN R, CARBON R (2003) Experience base schema building blocks of the PLEASERS library. *Journal of Universal Computer Science* 9(7), 659–669.

(Fowler, 1997) FOWLER M (1997) *Analysis Patterns: reusable object models*. Addison-Wesley, ISBN: 978-0201895421

(Fowler, 2003) FOWLER M (2003) Patterns. *IEEE Software* 20(2), 56–57, ISBN: 0-321-12742-0

(Freeze & Kulkarni, 2011) FREEZE R, KULKARNI U (2011) Understanding the Composition of Knowledge Management Capability. *Organizational Learning and Knowledge: Concepts, Methodologies, Tools and Applications (4 Vol)*, 208.

(Gamma, 1995) GAMMA E, JOHNSON R, HELM R, VLISSIDES J (1995) Design patterns: Elements of reusable object-oriented software. *Addison-Wesley*, ISBN: 978-0201633610

(Garcia & Turner, 2006) GARCIA S, TURNER R (2006) *CMMI® survival guide: just enough process improvement*. Addison-Wesley Professional, ISBN: 978-0-321-49178-7

(García et al., 2012) GARCÍA GUZMÁN J, MARTÍN D, URBANO J, AMESCUA A (2012) Practical experiences in modelling software engineering practices: The project patterns approach. *SOFTWARE QUALITY JOURNAL*, 1–30, DOI: 10.1007/s11219-012-9177-8

(García et al., 2011) GARCÍA J, AMESCUA A, SÁNCHEZ M-I, BERMÓN L (2011) Design guidelines for software processes knowledge repository development. *Information and Software Technology* 53(8), 834–850, DOI: 10.1016/j.infsof.2011.03.002.

(Gnatz et al., 2003) GNATZ M, MARSCHALL F, POPP G, RAUSCH A, SCHWERIN W (2003) The living software development process. *Software Quality Professional* 5(3), 4–16.

(Group Processworks, 2011) GROUP PROCESSWORKS (2011) *EzyLib Process Asset Libraries*, URL: http://www.processworksgroup.com.

(Herbsleb et al., 1994) HERBSLEB J, CARLETON A, ROZUM J, SIEGEL J, ZUBROW D (1994), "Benefits of CMM-based software process improvement: Initial results," 1994, available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.8646&rep=rep1&type=pdf.

(Iida, 1999) IIDA H (1999) Pattern-Oriented Approach to Software Process Evolution. *Workshop on the Principles of Software Evolution*.

(Isla-Montes, 2004) ISLA-MONTES JL, GUTIÉRREZ-VELA FL (2004) Diseño en base a patrones . Aplicación a sistemas.

(Ivarsson & Gorschek, 2011) IVARSSON M, GORSCHEK T (2011) Tool support for disseminating and improving development practices. *Software Quality Journal*, 1–27, DOI: 10.1007/s11219-011-9139-6

(Jacobson et al., 2007) JACOBSON I, WEI P, SPENCE I (2007) Enough of Processes-Lets do Practices. *Journal of Object Technology* 6(6), 41–66.

(Jalote, 2002) JALOTE P (2002) *Software project management in practice*. Addison-Wesley Longman Publishing Co., Inc., ISBN: 0-201-73721-3

(Jennex, 2004) JENNEX ME, OLFMAN L (2004) Modeling knowledge management success. Proceedings of the Conference on Information Science and Technology Management.

(Johnson et al., 2010) JOHNSON AR, GAMMA E, VLISSIDES J, HELM R (2010) Design Patterns : Elements of Reusable Object-Oriented Software. *Science*, 2010–2010.

(Juristo & Moreno, 2001) JURISTO N, MORENO AM (2001) *Basics of software engineering experimentation*.Kluwer Academic Publishers, ISBN: 0-7923-7990-X

(Kellner et al., 1998) KELLNER MI, BECKER-KORNSTAEDT U, RIDDLE WE, TOMAL J, VERLAGE M (1998) Process guides: effective guidance for process participants. *Proceedings of the Fifth International Conference on the Software Process*(June), 11–25.

(Landaeta et al., 2008) LANDAETA JF, GARCÍA J, AMESCUA A (2008) Practical SPI Planning. *Software Process Improvement* 16(3), 82–93, DOI: 10.1007/978-3-540-85936-9_8

(Layman, 2005) LAYMAN B (2005) Implementing an Organizational Software Process Improvement Program. *IEEE Software Engineering* 2, 279–288.

(Lindsey, 2002) LINDSEY K (2002) Measuring knowledge management effectiveness : A Task-Contingent organizational capabilities perspective. Proceedings of the Eighth Americas Conference on Information Systems p 2085 – 2090.

(Llorens et al., 2004) LLORENS J, MORATO J, GENOVA G (2004) RSHP: an information representation model based on relationships. *Studies in fuzziness and soft computing* 159, 221–253, Ed. Damiani, Ernesto; Jain, Lakhmi C.; Madravio, Mauro. Springer Verlag, New Cork. ISBN: 3-540-22030-5

(Lukosch & Schümmer, 2006) LUKOSCH S, SCHUMMER T (2006) Groupware development support with technology patterns. *International Journal of Human-Computer Studies* 64(7), 599–610, DOI: 10.1016/j.ijhcs.2006.02.006

(Lukosch & Schümmer, 2004) LUKOSCH S, SCHÜMMER T (2004) Communicating design knowledge with groupware technology patterns. *Groupware: Design, Implementation and Use*, 223–237.

(Maier, 2007) MAIER R (2007) *Knowledge management systems: Information and communication technologies for knowledge management.*Springer-Verlag, ISBN: 3540714073

(Martinez et al., 2005) MARTINEZ P, AMESCUA A, GARCIA J, CUADRA D, LLORENS J, FUENTES JM, MARTÍN D, CUEVAS G, CALVO-MANZANO JA, FELIU TS (2005) Requirements for a knowledge management framework to be used in software intensive organizations. *Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference* p 554–559.

(Martín et al., 2012a) MARTÍN D, GARCÍA GUZMÁN J, URBANO J, AMESCUA A (2012a) Modelling Software Development Practices using Reusable Project Patterns: A Case Study. *Journal of Software: Evolution and Process*.

(Martín et al., 2012b) MARTÍN D, GARCÍA GUZMÁN J, URBANO J, LLORENS J (2012b) Patterns as objects to manage knowledge in software development organizations. *Knowledge Management Research & Practice* 10(3), 252–274, DOI:10.1057/kmrp.2012.15.

(Martín et al., 2011a) MARTÍN D, GARCÍA J, URBANO J, AMESCUA A (2011a) Modeling Software Development Practices using Reusable Project Patterns: A Case Study. *EuroSPI 2011 Industrial Proceedings* p 5.1–5.10 DELTA, Denmark.

(Martín et al., 2011b) MARTÍN D, MARRERO M, URBANO J, BARRA E, MOREIRO JA (2011) Virtualización: Una Solución para la Eficiencia, Seguridad y Administración de Intranets. *El Profesional de la Información* 20(3), 348–354, DOI:10.3145/epi.2011.may.16

(Martín et al., 2007) MARTÍN D, GARCÍA J, AMESCUA A, LLORENS J (2007) Reusable Project Patterns to enhance Software Process Improvement. *EuroSPI 2007 Industrial Proceedings* p 3.25–3.34.

(Massey et al., 2002) MASSEY A, MONTOYA-WEISS M, O'DRISCOLL T (2002) Knowledge Management in Pursuit of Performance: Insights from Nortel Networks. *Management Information Systems Quarterly* 26(3), 269–289.

(Maurer & Holz, 2002) MAURER F, HOLZ H (2002) Integrating process support and knowledge management for virtual software development teams. *Annals of Software Engineering* 14(1), 145–168, DOI: 10.1023/A:1020505708326

(Medina-Dominguez, 2010) MEDINA-DOMINGUEZ F "Marco Metodológico para la Mejora de la Eficiencia de Uso de los Procesos Software," 2010.

(Medina-Dominguez et al., 2010) MEDINA-DOMINGUEZ F, SANCHEZ-SEGURA M-I, MORA-SOTO A, AMESCUA A (2010) Reverse Engineering and Software Products Reuse to Teach Collaborative Web Portals: A Case Study With Final-Year Computer Science Students. *IEEE Transactions on Education* 53(4), 595–607, DOI: 10.1109/TE.2009.2037313

(Mentzas et al., 2001) MENTZAS G, APOSTOLOU D, YOUNG R, ABECKER A (2001) Knowledge networking: a holistic solution for leveraging corporate knowledge. *Journal of Knowledge Management* 5(1), 94–107.

(Merrill, 2000) MERRILL MD (2000) Knowledge objects and mental models. Proceedings International Workshop on Advanced Learning Technologies. IWALT 2000. Advanced Learning Technology: Design and Development Issues p 244–246 IEEE Comput. Soc., DOI: 10.1109/IWALT.2000.890621

(Microsoft Corporation, 2010) MICROSOFT CORPORATION (2010) *Visual Studio 2010 Ultimate*, URL: http://www.microsoft.com/visualstudio/en-us/products/2010-editions/ultimate.

(Moe & Dybå, 2006) MOE NB, DYBÅ T (2006) The use of an electronic process guide in a medium-sized software development company. *Software Process: Improvement and Practice* 11(1), 21–34, DOI: 10.1002/spip.250.

(Mora-Soto et al., 2010) MORA-SOTO A, SANCHEZ-SEGURA M-I, MEDINA-DOMINGUEZ F, AMESCUA A (2010) Transactive Memory System Proposal to Foster Collaborative Learning and Knowledge Sharing into Organizations. International Conference on Computer Supported Education (CSEDU 2010).

(Nodder & Nielsen, 2008) NODDER C, NIELSEN J (2008) *Agile usability: best practices for user experience on agile development projects*. Nielsen Norman Group.

(Nonaka, 2006) NONAKA I (2006) Organizational Knowledge Creation Theory: Evolutionary Paths and Future Advances. *Organization Studies* 27(8), 1179–1208, DOI: 10.1177/0170840606066312.

(Nonaka & Konno, 1999) NONAKA I, KONNO N (1999) The concept of 'Ba': building a foundation for knowledge creation. *The Knowledge Management Yearbook 1999-2000*, 37.

(Nonaka & Takeuchi, 1995) NONAKA I, TAKEUCHI H (1995) *The knowledge-creating company: how Japanese companies create the dynamics of innovation, ISBN: 0195092694*

(OMG, 2007) Object Management Group (2007) *OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2*, URL:http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF.

(OMG, 2008) Object Management Group (2008) *Software & Systems Process Engineering Meta-Model Specification*. URL: http://www.omg.org/spec/SPEM/2.0/

(Osellus, 2007) OSELLUS (2007) *IRIS Process Author*, URL: http://www.osellus.com/IRIS-PA.

(Paulk et al., 1993) PAULK M, WEBER C, GARCIA S, CHRISSIS M, BUSH M (1993), "Key Practices of the Capability Maturity Model, Version 1.1. Software Engineering Institute," 1993.

(Petter & Vaishnavi, 2007) PETTER S, VAISHNAVI V (2007) Facilitating Experience Reuse among Software Project Managers. *Information Sciences* 178(7), 19, DOI: 10.1016/j.ins.2007.11.020.

(Rowley, 2007) ROWLEY J (2007) The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science* 33(2), 163–180, DOI: 10.1177/0165551506070706.

(Rus & Lindvall, 2002) RUS I, LINDVALL M (2002) Knowledge management in software engineering. *Software, IEEE* 19(3), 26–38, DOI: 10.1109/MS.2002.1003450.

(Sanchez-Segura et al., 2010) SANCHEZ-SEGURA M, MEDINA-DOMINGUEZ F, AMESCUA A DE, MORA-SOTO A (2010) Improving the efficiency of use of software engineering practices using product patterns. *Information Sciences* 180(14), 2721–2742, DOI: 10.1016/j.ins.2010.03.028.

(Scholz & Tietje, 2002) SCHOLZ RW, TIETJE O (2002) *Embedded case study methods: Integrating quantitative and qualitative knowledge*. Sage Publications, Inc,.

(Scott et al., 2002) SCOTT L, CARVALHO L, JEFFERY R (2002) A process-centred experience repository for a small software organisation. *Ninth Asia-Pacific Software Engineering Conference, 2002.*(02), 603–609, DOI: 10.1109/APSEC.2002.1183096.

(SEI, 2010) Software Engineering Institute (2010), "CMMI for Development. Version 1.3 Technical Report (CMU/SEI-2010-TR-033)," 2010, available at: http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.

(Select Business Solutions, 2011) SELECT BUSINESS SOLUTIONS (2011) *Select Process Director*, URL: http://www.selectbs.com/process-improvement/select-process-director.

(Sheard, 2003) SHEARD SA (2003) Process Implementation. Proceeding of the 13th Annual Symposium of the International Council on Systems Engineering, Arlington, VA p 1114–1124.

(Sommerville, 2004) SOMMERVILLE I (2004) *Software Engineering (7th Edition)*. Addison Pearson Education, ISBN: 9788177585308

(Standish Group, 2004) STANDISH GROUP (2004) , "2004 THIRD QUARTER RESEARCH REPORT," 2004, available at: http://blog.nalis.fr/public/pdf/q3-spotlight.pdf.

(Stoyko et al., 2007) STOYKO P, FANG Y, GUIMONT F (2007) *Lost & Found: A Smart-practice Guide to Managing Organizational Memory*. Canada School of Public Service.

(Surhone et al., 2010) SURHONE LM, TENNOE MT, HENSSONOW SF (2010) *Software Analysis Pattern*. VDM Verlag, ISBN: 9786133077218.

(The Eclipse Foundation, 2011) THE ECLIPSE FOUNDATION (2011) *Eclipse Process Framework Composer*, URL: http://www.eclipse.org/epf/general/description.php.

(Verma & Tiwari, 2009) VERMA A, TIWARI MK (2009) Role of corporate memory in the global supply chain environment. *International Journal of Production Research* 47(19), 5311–5342, DOI: 10.1080/00207540801918570.

(Visaggio, 2009) VISAGGIO G (2009) Knowledge Base and Experience Factory for Empowering Competitiveness. *Software Engineering, LNCS 2009* 5413/2009, 223–256, DOI: 10.1007/978-3-540-95888-8_9.

(Yin, 2009) YIN RK (2009) *Case study research: Design and methods*. Sage Publications, Inc.

(Zack, 2002) ZACK MH (2002) Developing a knowledge strategy. *The strategic management of intellectual capital and organizational knowledge*, 255–276.

ANEXOS

Tabla de contenidos: Anexos

7: ANEXOS

7.1 Méritos

7.1.1 Artículos de journal indexados en el JCR, relacionados con esta tesis doctoral.

Título:	**Practical Experiences in Modelling Software Engineering Practices: The Project Patterns Approach**
Autores:	Javier García Guzmán, Diego Martín, Julián Urbano, Antonio Amescua
Journal:	Software Quality Journal (2012)
Factor de impacto:	0.75 (2010) (García et al., 2012)

Título:	**Patterns as Objects to Manage Knowledge in Software Development Organizations**
Autores:	Diego Martín, Javier García Guzmán, Julián Urbano, Juan Lloréns
Journal:	Knowledge Management Research & Practice (2012)
Factor de impacto:	0.855 (2010) (Martín et al., 2012b)

Título:	**Modelling Software Development Practices using Reusable Project Patterns: A Case Study**
Autores:	Diego Martín, Javier García Guzmán, Julián Urbano, Antonio Amescua
Journal:	Journal of Software: Evolution and Process (2012)
Factor de impacto:	0.844 (2010) (Martín et al., 2012a)

7.1.2 Artículos de journal indexados en el JCR, no relacionados con esta tesis doctoral

Título:	**Virtualización: Una Solución para la Eficiencia, Seguridad y Administración de Intranets**
Autores:	Diego Martín, Mónica Marrero, Julián Urbano, Eduardo Barra, José Antonio Moreiro
Journal:	El Profesional de la Información (2011)
Factor de impacto:	0.478 (2009) (Martín et al., 2011b)

7.1.3 Participación en proyectos asociados a esta tesis doctoral.

Código	**2004/02925/00**
Título	GPS: Plataforma de gestión de procesos software: Modelado, reutilización y medición.
Investigador principal	PALOMA MARTINEZ FERNANDEZ
Financiador	MINISTERIO DE EDUCACION Y CIENCIA DIR. GRAL. INVESTIGACION
Fecha de inicio	12/12/2004
Fecha de fin	11/06/2008

7.1.4 Participación en proyectos no asociados a esta tesis doctoral.

Código	**2008/00086/00**
Título	CP07: COMBINACIÓN DE TÉCNICAS NO CONVENCIONALES APLICADAS A LA EXTRACCIÓN DE INFORMACIÓN
Investigador principal	JOSE ANTONIO MOREIRO GONZALEZ
Financiador	COMUNIDAD DE MADRID-UC3M
Fecha de inicio	01/01/2008
Fecha de fin	28/02/2009

Código	**2007/04055/00**
Título	CP06: DESARROLLO DE UN SISTEMA DE INTELIGENCIA ESTRATÉGICA BASADO EN LA GESTIÓN SEMÁNTICA DE CONOCIMIENTO
Investigador principal	JOSE ANTONIO MOREIRO GONZALEZ
Financiador	COMUNIDAD DE MADRID-UC3M
Fecha de inicio	01/01/2007
Fecha de fin	31/03/2008

Código	**2007/04332/00**
Título	Desarrollo de un sistema de recuperación conceptual mediante niveles semánticos en la representación de esquemas metadatos
Investigador principal	JUAN BAUTISTA LLORENS MORILLO
Financiador	MINISTERIO DE EDUCACION Y CIENCIA SEC. DE ESTADO DE UNIVERSIDADES E INVEST.
Fecha de inicio	01/10/2007
Fecha de fin	31/12/2010

Código	**2006/03528/00**
Título	CP05-SISTEMA DE REPRESENTACIÓN Y RECUPERACIÓN SEMÁNTICA DE ESQUEMAS BASADO EN ONTOLOGÍAS MULTINIVEL
Investigador principal	JOSE ANTONIO MOREIRO GONZALEZ
Financiador	COMUNIDAD DE MADRID-UC3M
Fecha de inicio	01/01/2006
Fecha de fin	31/12/2006

7.1.5 Contribuciones a conferencias relacionadas con esta tesis doctoral

Título:	**Requirements for a knowledge management framework to be used in software intensive organizations**
Autores:	Paloma Martinez, Antonio Amescua, Javier Garcia, Dolores Cuadra, Juan Llorens, J.M. Fuentes, Diego Martín, Gonzalo Cuevas, J.A. Calvo-Manzano, and Tomas S. Feliu
Conferencia:	Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference

Título:	**A Pattern-Based Approach to Deploy Process Improvements in Small Settings**
Autores:	Antonio Amescua , Gonzalo Cuevas , Javier García Guzmán, Juan Llorens, Paloma Martínez, and Diego Martín
Conferencia:	First International Research Workshop for Process Improvement in Small Settings 2005

Título:	**Reusable Project Patterns to enhance Software Process Improvement**
Autores:	Diego Martín, Javier García Guzman, Antonio Amescua, and Juan Llorens
Conferencia:	European Systems and Software Process Improvement and Innovation Conference (EuroSPI'2007)

Título:	**Modeling Software Development Practices using Reusable Project Patterns: A Case Study**
Autores:	Diego Martín, Javier García, Julián Urbano, and Antonio Amescua
Conferencia:	European Systems and Software Process Improvement and Innovation Conference (EuroSPI'2011)

7.1.6 Contribuciones a conferencias no relacionadas con esta tesis doctoral

Título:	**PARTICULARIZACIÓN DINÁMICA DE PLANTILLAS PARA LA GESTIÓN DE REQUISITOS**
Autores:	Omar Hurtado, Mario Quinde, and Diego Martín
Conferencia:	V Congreso Internacional de Ingeniería de Sistemas ICSE 2009, 2009

Título:	**Crowdsourcing Preference Judgments for Evaluation of Music Similarity Tasks**
Autores:	Julián Urbano, Jorge Morato, Mónica Marrero, and Diego Martín
Conferencia:	ACM SIGIR workshop on Crowdsourcing for Search Evaluation, 2010

Título:	**Improving the Generation of Ground Truths based on Partially Ordered Lists**
Autores:	Julián Urbano, Mónica Marrero, Diego Martín, and Juan Lloréns
Conferencia:	International Society for Music Information Retrieval Conference, 2010

Título:	**Bringing Undergraduate Students Closer to a Real-World Information Retrieval Setting: Methodology and Resources**
Autores:	Julián Urbano, Mónica Marrero, Diego Martín, and Jorge Morato
Conferencia:	ACM SIGCSE Annual Conference on Innovation and Technology in Computer Science Education, 2011

Título:	**Audio Music Similarity and Retrieval: Evaluation Power and Stability**
Autores:	Julián Urbano, Diego Martín, Mónica Marrero and Jorge Morato
Conferencia:	International Society for Music Information Retrieval Conference, 2011

Título:	**Overview of EIREX 2012: Social Media**
Autores:	Julián Urbano, Mónica Marrero, Diego Martín and Jorge Morato
Conferencia:	ACM Computing Research Repository, 2012

Título:	**Overview of EIREX 2011: Social Media**
Autores:	Julián Urbano, Diego Martín, Mónica Marrero and Jorge Morato
Conferencia:	ACM Computing Research Repository, 2011

Título:	**The University Carlos III of Madrid at TREC 2011 Crowdsourcing Track**
Autores:	Julián Urbano, Mónica Marrero, Diego Martín, Karina Robles and Juan Lloréns
Conferencia:	Text REtrieval Conference, 2011

Título:	**Overview of EIREX 2010: Computing**
Autores:	Julián Urbano, Mónica Marrero, Diego Martín, Jorge Morato
Conferencia:	ACM Computing Research Repository, 2010

Título:	**The University Carlos III of Madrid at TREC 2011 Crowdsourcing Track: Notebook Paper**
Autores:	Julián Urbano, Mónica Marrero, Diego Martín, Karina Robles and Juan Lloréns
Conferencia:	Text REtrieval Conference Notebook, 2011

7.1.7 Otras publicaciones

Título:	**Diseño basado en componentes**
Autores:	Diego Martín, Juan Llorens, Anabel Fraga
Publicación:	Open Course Ware
Fecha	Diciembre 2009
URL	http://ocw.uc3m.es/ingenieria-informatica/diseno-basado-en-componentes

Título:	**Components Based Software Design**
Autores:	Juan Llorens, Anabel Fraga, Diego Martín
Publicación:	Open Course Ware
	Julio 2010
URL	http://ocw.uc3m.es/ingenieria-informatica/components-based-software-design

7.2 Ejemplos de sdPP

7.2.1 Ejemplo sdPP "Tipo 1"

7.2.1.1 Descripción

Propuesta por Kent Becket en 1996 la programación extrema es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Esta metodología tiene como principio basarse en los valores de simplicidad, comunicación, retroalimentación y valor.

Tipo de metodología: Evolutiva, Incremental e Iterativa.

7.2.1.2 WBS

1. Iteración 0
   1.1. Inicio del proyecto
2. Iteraciones construcción
   2.1. Seleccionar los elementos de trabajo de mayor prioridad
   2.2. Iteración
      2.2.1. Reunión diaria "de pie"
      2.2.2. Desarrollo diario
      2.2.3. Pruebas independientes
   2.3. Revisión de iteración
   2.4. Retrospectiva
3. Entrega

3.1. Llevar el sistema a producción

4. Producción

4.1. Operar y mantener el sistema en producción

5. Retiro del sistema

5.1. Quitar el sistema de producción

7.2.1.3 Workflow

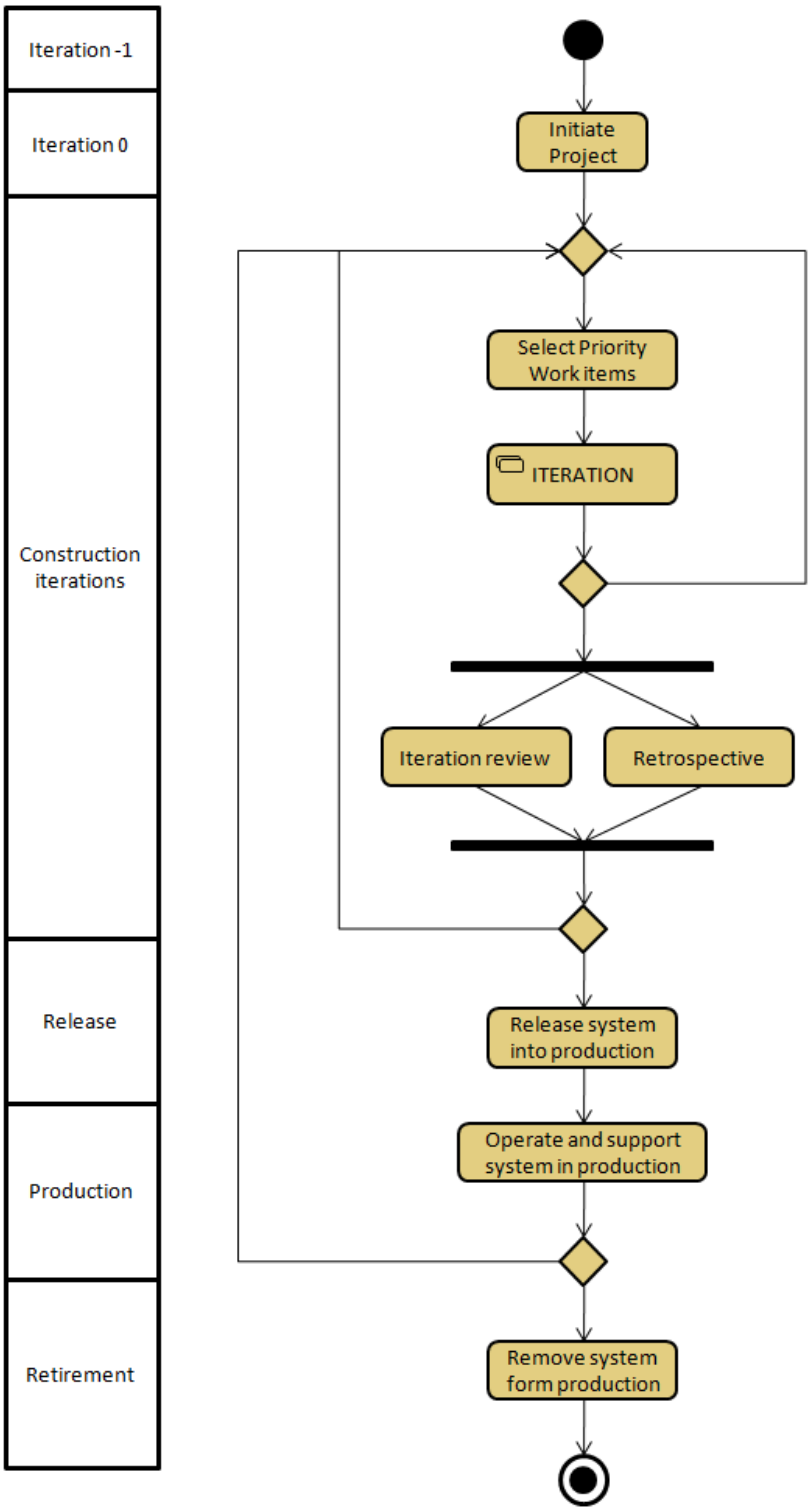


Figura 59 Workflow de XP

Dentro de la actividad iteración:

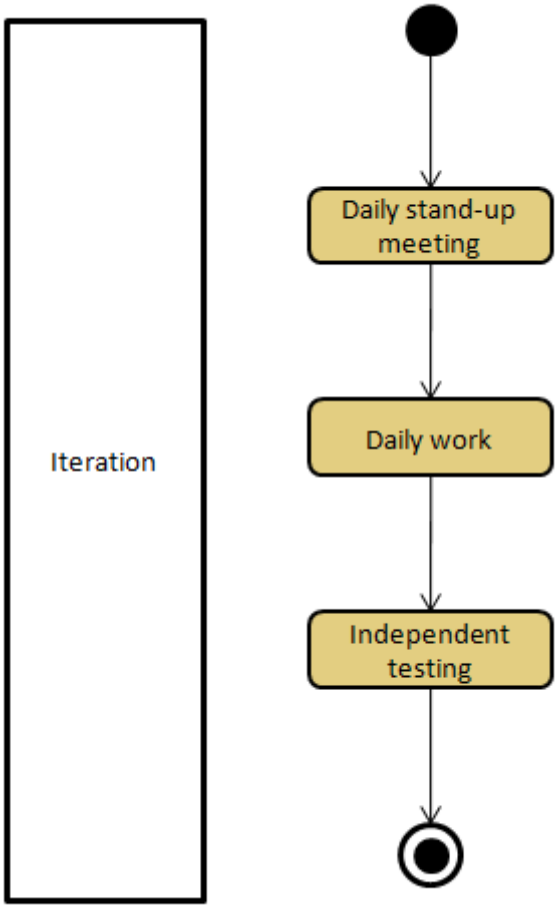


Figura 60 Workflow en detalle de la actividad "Iteración" de XP

7.2.1.4 Productflow

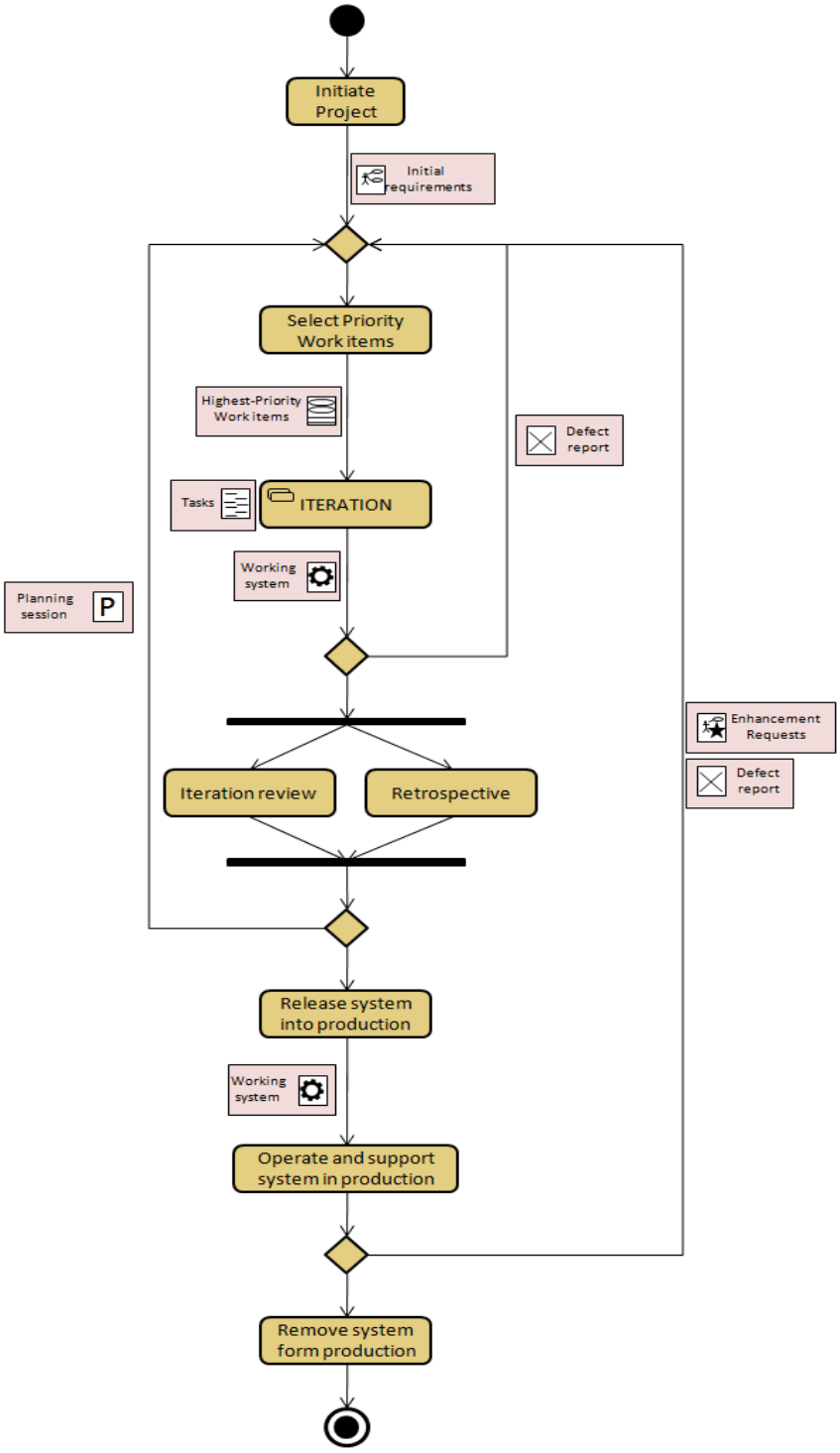


Figura 61 Productflow de XP

7.2.1.5 Productos

Los productos que podemos encontrar en la metodología de desarrollo XP son:

- Requisitos Iniciales de Arquitectura.
- Historias de Usuario.
- Tareas – CRC.
- Release resultante de la Iteración.
- Planning: Nueva Planificación resultante de la Iteración.
- Informe de Defectos del Sistema.

7.2.1.6 Plantillas

Para la metodología XP se han creado las siguientes plantillas para facilitar la labor de los productos:

Historias de Usuario

ID	
Nombre	
Modificación de HU	
Usuario	
Iteración Asignada	
Prioridad en negocio	
Riesgo en Desarrollo	
Puntos Estimados	
Puntos Reales	
Descripción	
Observaciones	

Figura 62 Plantilla para las Historias de Usuario en XP

Las historias de usuario sirven para el mismo propósito que los casos de uso, pero no son iguales. Se utilizan para definir los requerimientos de un sistema de software, y también para crear estimaciones para la planificación de las iteraciones. Las historias de usuario son escritas por los clientes en forma de las cosas que quieren que el sistema haga por ellos. Son similares a los escenarios de uso, pero no se limitan solamente a la interfaz de usuario.

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, remplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. A continuación puede verse un ejemplo.

Historia de Usuario	
Número: 3	**Nombre:** Elaboración de Presupuesto
Usuario: Desarrollo Empresarial	
Modificación de Historia Nº: 0	**Iteración Asignada: 2**
Prioridad en Negocio: Media **(Alta/Media/Baja)**	**Puntos estimados:**
Desarrollador Encargado: Luis Ariel Castillo Estupiñan	
Descripción: Elaborar el presupuesto para la ejecución del evento. Depende de la disponibilidad presupuestal estipulada para la dependencia para el año vigente.	
Observaciones: El presupuesto está estipulado para la dependencia para el año, lo cual implica que para cada evento se tiene un máximo de recursos dependiendo de la naturaleza del mismo.	

Figura 63 Ejemplo de Historia de Usuario en XP

Tarjeta de Tarea

ID	
Nombre	
Historia de Usuario	
Programador Responsable	
Tipo Tarea	
Fecha inicio	
Fecha Fin	
Puntos Estimados	
Descripción	

Figura 64 Plantilla para las Tarjetas de Tarea en XP

Tarjetas CRC

Nombre de la clase	
Responsabilidades	
Colaboradores	

Figura 65 Plantilla para las tarjetas CRC en XP

Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

En la práctica conviene tener pequeñas tarjetas de cartón, que se llenarán y que son mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas.

Los pasos a seguir para llenar las tarjetas son los siguientes:

- Encontrar clases.
- Encontrar responsabilidades.
- Definir colaboradores.
- Disponer las tarjetas.

Para encontrar las clases debemos pensar qué cosas interactúan con el sistema (en nuestro caso el usuario), y qué cosas son parte del sistema, así como las pantallas útiles a la aplicación (un despliegue de datos, una entrada de parámetros y una pantalla general, entre otros).

Una vez que las clases principales han sido encontradas se procede a buscar los atributos y las responsabilidades y finalmente se buscan los colaboradores dentro de la lista de clases que se tenga.

7.2.1.7 To dos

- La planificación: se utilizan las user-stories (historias de usuario), para realizar el análisis. Se dividirán en tareas priorizadas y cada una de ellas tendrá un desarrollo incremental.
- Versiones pequeñas: La primera versión contendrá el conjunto mínimo de requisitos más útiles/necesarios para el sistema global.
- Sistema metafórico: Cada proyecto debe tener una metáfora asociada que nos ofrezca unos criterios para nombrar lo que vayamos haciendo de forma fácil.
- Diseño simple: Cómo los requerimientos cambian, o pueden hacerlo, diariamente, hay que utilizar los diseños más simples posibles para cumplir los requerimientos que tenemos en la actualidad.
- Testeo continuo: Antes de que se implemente cualquier característica de un sistema, se debe escribir un test para ella.
- Refactoring: Cuando tenemos que introducir una nueva característica del sistema, si esta tiene mucho en común con otra previa, lo mejor es eliminar el código duplicado, sin miedo a que falle, debido a que el test probará el correcto funcionamiento.
- Pair programming (programación en parejas): Se trabaja en parejas, cada una utilizando un único ordenador. Así, el código se revisa mientras se desarrolla.
- Propiedad colectiva del código: Cualquiera puede modificar cualquier módulo en cualquier momento, nadie tiene la propiedad de ningún módulo.
- Integración continua: Todos los cambios se introducen en el sistema, al menos, una vez al día.
- Semanas de 40 horas de trabajo: Los programadores se deben ir a casa a su hora.

- Cliente en su sitio: Siempre hay un usuario del sistema que es accesible por los miembros del equipo de trabajo.
- Estándares de codificación: Todos deben usar los mismos criterios a la hora de programar. De esta forma, no sería posible determinar quién ha realizado una determinada parte de la implementación.

7.2.1.8 Not To Dos

- Realizar fragmentos de código duplicados.
- Comenzar una nueva iteración sin la aprobación del cliente.
- Los desarrolladores no trabajarán en habitaciones separadas.
- Ningún cliente en el equipo.
- XP es exclusivamente desarrollo iterativo + documentación mínima + pruebas unitarias.
- No escribir las pruebas unitarias antes de codificar.
- El cliente no decide.
- No pruebas de aceptación en cada iteración.
- Rediseño mínimo.
- Tener exclusivamente un único cliente en el equipo.
- Muchas tareas de grado excesivamente fino.
- Emparejamiento con un compañero mucho tiempo.
- No integrar el equipo de aseguramiento de la calidad en el proyecto.
- Escribir la documentación de diseño después de codificar.
- Mucho modelado es malo.
- Solo utilizar programación en parejas con jóvenes.
- Emparejamiento de noveles.
- El observador no puede ver fácilmente el monitor.
- Equipo que no conoce XP ni sus motivaciones.
- Disensiones en el equipo.
- Reuniones periódicas demasiado largas y sin objetivos claros.
- La no utilización de un probador dedicado.
- El cliente y el gran jefe no están alineados.

- El cliente que escribe las pruebas de aceptación no es el mismo que las ejecuta.
- Iteraciones demasiado largas.
- Las iteraciones no son time-boxed.
- La iteración no finaliza en una línea base perfectamente integrada.
- Cada iteración finaliza en una entrega al usuario.
- Planificación predictiva.

7.2.1.9 Metadatos

Backup Fiable:	NA	Actualización Online:	NA
Comunicaciones de Datos:	NA	Interfaz compleja:	1
Funciones distribuidas:	NA	Procesamiento complejo:	1
Rendimiento:	4	Reusabilidad:	3
Configuración muy usada:	3	Fácil Instalación:	NA
Entrada de datos online:	NA	Múltiples Sites:	0
Fácil de operar:	4	Facilita cambios:	5

7.2.1.10 Requisitos

- Se debe tener un sistema que ayude a la gestión de la metodología
- Se debe educar a la plantilla para poder llevar un desarrollo con XP
- La plantilla debe saber trabajar en grupo porque el desarrollo se hace en parejas.
- Equipo acostumbrado a alta velocidad de trabajo
- La plantilla debe estar acostumbrada a cumplir un horario de 40 horas semanales.
- Es necesario tener una sala común para poder hacer reuniones.
- Todos los desarrolladores deben estar en una única habitación.
- En la habitación debe haber comida (básicamente "snacks para reforzamiento positivo").
- Todos los días comienzan con una reunión de apertura.

7.2.1.11 Riesgos

- No separar las parejas de programación para intentar producir más, ya que corromperíamos un requisito de XP.
- Si se cuenta con una plantilla poco entrenada en XP se puede sufrir el riesgo de fallar.
- No hay que sentirse tentado con la posibilidad de realizar más de 40 horas semanales de trabajo.