UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

GENERACIÓN DE PLANES DE DESARROLLO A PARTIR DE PROCESOS FORMALIZADOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

AGUSTÍN ANDRÉS LÓPEZ QUEVEDO

PROFESOR GUÍA:
DANIEL PEROVICH GEROSA

MIEMBROS DE LA COMISIÓN:
ÉRIC TANTER
MARÍA CECILIA BASTARRICA PIÑEYRO

SANTIAGO DE CHILE
2016

Resumen

El contexto de este trabajo de título se enmarca en empresas que pertenecen al rubro de desarrollo de software en Chile. En particular, este trabajo se enfoca en la operación de las pequeñas y medianas empresas (PyMEs). Estas son típicamente diferentes a las grandes empresas de software en términos de su operatoria, lo cual hace que la gestión de los proyectos requiera consideraciones particulares para ellas.

Un problema de las PyMEs de desarrollo de software chilenas es que carecen de un proceso automático para la generación de planes de desarrollo a partir de sus procesos formalizados. Es decir, a partir de sus procesos formalizados ellos tienen que crear manualmente sus planes de desarrollo, lo que les toma tiempo y esfuerzo considerable.

El objetivo general de la memoria es el definir, diseñar e implementar una herramienta capaz de generar, en forma automática, planes de desarrollo para proyectos de software a partir del proceso de desarrollo de una empresa y de la información de los recursos asignados al proyecto.

En este proyecto se implementó un sistema capaz de obtener la información de un proceso formalizado, para luego generar de forma automática un plan de desarrollo. Para ello se definieron los requisitos mínimos que el sistema debía cumplir; dentro de los más destacables, se emplean tecnologías open source y las más populares en el rubro. Las herramientas utilizadas fueron: Eclipse Process Framework Composer (EPFC) en la cual se formalizan los procesos de desarrollo de las PyMEs y Redmine, una plataforma web de gestión de proyectos, usada para crear planes de desarrollo.

La solución consiste en dos plugins y un artefacto de intercambio. Uno de ellos llamado `XmlGen` se integra a EPFC y extrae la información del proceso formalizado y la transfiere a un artefacto de intercambio, que en este caso es un archivo XML. Finalmente el segundo plugin, llamado `XmlImport`, se integra con Redmine, éste lee la información del archivo XML y construye el plan de desarrollo del proyecto en la plataforma.

Con el fin de validar la solución propuesta y desarrollada, se utilizó un proceso de desarrollo de software formalizado por la PyME de desarrollo de software Ki Teknology. Con él se presentó una serie de demostraciones del sistema a paneles de expertos que opinaron sobre el funcionamiento de este, lo que conllevó a su validación y, finalmente, a un potencial uso en la operación de las PyMEs. Finalmente se concluye que el sistema desarrollado logra satisfacer el problema planteado: generar automáticamente planes de desarrollo para un proyecto a partir de un proceso formalizado.

Agradecimientos

Me gustaría agradecer a mi profesor Daniel Perovich por su guía y apoyo a lo largo de este proceso. Así también, agradezco el apoyo de los integrantes de la comisión, del equipo del proyecto GEMS y finalmente a la empresa Ki Teknology ya que contribuyeron de algún u otra forma a la realización de esta memoria.

Por último, doy gracias a mi familia y amigos por su constante apoyo, que no solo se limita a la realización de este trabajo, si no que va mucho más allá.

Tabla de contenido

Índice de figuras

Índice de tablas

1 Introducción

El Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile ha llevado adelante sucesivos proyectos de investigación dirigidos a apoyar a las pequeñas y medianas empresas (PyMEs) de Chile dedicadas al desarrollo de software en la formalización, uso y mejora de sus procesos de desarrollo.

Un proceso de desarrollo de software es un conjunto parcialmente ordenado de actividades, los roles de las personas que las realizan y los artefactos que estos producen y utilizan al realizarlas, con el fin de crear y mantener un conjunto de entregables solicitados por un cliente [1]. Para una empresa de desarrollo, contar con un proceso bien definido es un factor determinante para alcanzar una mayor calidad en los productos de software y una mayor productividad en los proyectos de desarrollo [2]. Además, si el proceso definido está formalizado en una herramienta de especificación de procesos, la empresa puede construir o utilizar otras herramientas para la publicación, consulta y búsqueda de elementos del proceso; como así también para el análisis y mejora del mismo, y potencialmente, para la generación automática de otros documentos.

La formalización de los procesos requiere de habilidades de ingeniería de procesos y de software avanzadas y de un esfuerzo considerable. Es por ello que, en el contexto de los proyectos de investigación del DCC, se apoyó a seis PyMEs de software chilenas en la formalización de sus procesos de desarrollo. Al mismo tiempo y durante el transcurso de la formalización, se han diseñado y construido herramientas para capitalizar este esfuerzo. En una primera etapa, se construyó la herramienta AVISPA [3] que permite el análisis estático de los procesos y la detección de oportunidades de mejora. Con ella, las empresas logran mejorar sus procesos formalizados. En una segunda etapa, se construyó la herramienta de adaptación ADAPTE [4] que permite ajustar el proceso de desarrollo a las características específicas de un proyecto, obteniendo así un proceso adaptado al contexto del proyecto. Sin embargo, estas herramientas son de gran complejidad lo que ha limitado el impacto y uso de estas en las PyMEs de software chilenas.

Con el fin de solucionar el problema de adopción de las herramientas y de proveer un entorno integral para la mejora de procesos, nace en una tercera etapa el proyecto “Gestión Experimental de la Mejora de Software” (GEMS). Este proyecto propone integrar las herramientas desarrolladas previamente y aplicarlas en la práctica. Para ello GEMS propone complementar las herramientas actuales con nuevos desarrollos de software, para la ejecución de procesos, registro y recolección de métricas, visualización de información agregada, y soporte de herramientas y procedimientos para la mejora continua de la gestión de proyectos de desarrollo de software [5].

En el contexto de GEMS, uno de los objetivos es construir una herramienta que a partir del proceso adaptado de un proyecto específico, genere el plan de desarrollo. El resultado es el conjunto parcialmente ordenado de actividades que se ejecutarán en el proyecto, las fechas estimadas de inicio y fin, y las personas que están asignadas a los roles que participan y que ejecutan estas actividades.

1.1 Problema a abordar

Según una encuesta reciente a las PyMEs de software chilenas [6], más del 80% de las compañías encuestadas declaran tener su proceso definido y más de un 50% declaran haberlo formalizado. La encuesta mostró también que aquellas empresas con proceso definido tienen un porcentaje mayor de proyectos exitosos que las que no tienen proceso definido. Más del 50% de ellas declaran aplicar su proceso definido y casi el 40% indica que lo adaptan a las características del proyecto. En conclusión, la encuesta muestra que en las PyMEs de software chilenas se entiende la relevancia de tener procesos definidos, y que gran parte de ellas los definen y usan.

Una vez formalizados los procesos por las PyMEs de software, además de utilizarlos como repositorio de conocimiento y para la formación de personal nuevo, estos se utilizan para guiar la ejecución de los proyectos. Sin embargo, hoy en día, la aplicación de estas definiciones es principalmente referencial. Los jefes de proyecto planifican en base al proceso formalizado, pero esta actividad es realizada de forma principalmente manual y sin asistencia de herramientas que la automaticen. Las principales causas de esto son: (a) la empresa no cuenta o no conoce herramientas con esta funcionalidad, y (b) la adaptación del proceso a las características del proyecto es manual y por lo tanto, este debe procesarse previo a generar el plan de desarrollo.

Existen distintas aplicaciones para elaborar y gestionar planes de desarrollo de proyectos. Algunas de estas son: Microsoft Project, Redmine, OpenProject, entre otras. Cada PyME de software utiliza alguna de estas, o ha desarrollado una propia; como consecuencia es más difícil contar con una herramienta única que permita generar automáticamente planes de desarrollo y que se adecue al contexto de cualquier PyME.

Este trabajo de memoria propone el desarrollo de una herramienta que permita a un jefe de proyecto generar, en forma automática, los planes de desarrollo usando el proceso adaptado y la asignación de recursos de proyecto. Así, esta herramienta contribuye a las que están siendo desarrolladas en el proyecto GEMS. En la Figura 1 se muestran los artefactos, usuarios y herramientas involucrados en la formalización de procesos y en la generación del plan de desarrollo.

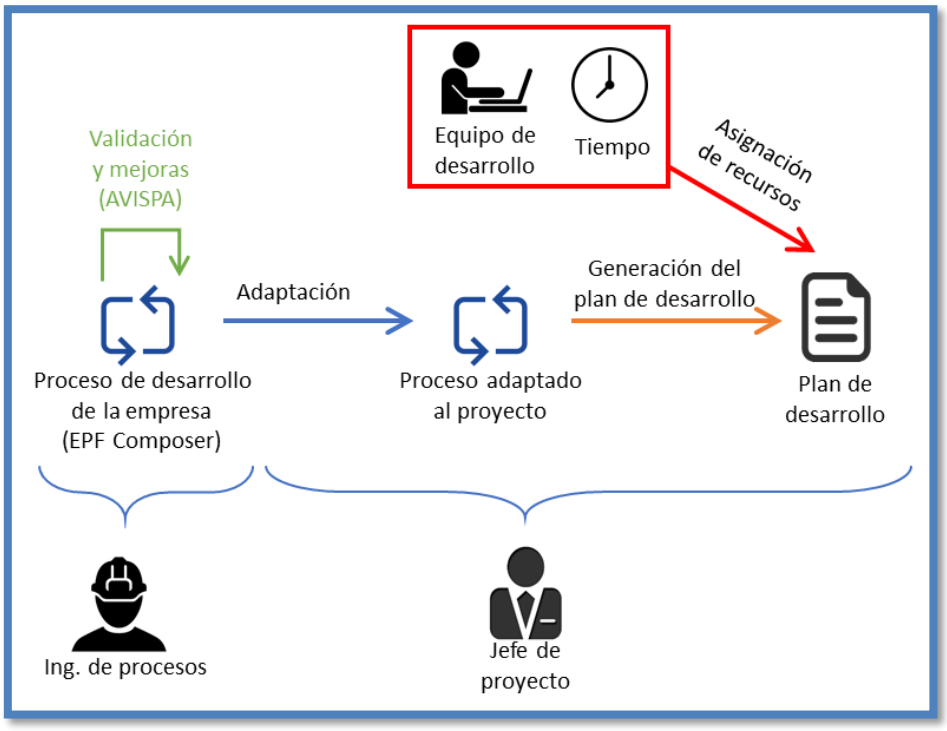


Figura 1: Flujo de información del proceso de generación de un plan de desarrollo

Por último, dado que la generación de planes de desarrollo es principalmente manual, no solo se está sub-utilizando la información disponible en el proceso formalizado, sino que se puede incurrir en errores que dificultan tanto la ejecución como el aprendizaje de este. Por ejemplo, la empresa PowerData, apoyada por el proyecto GEMS, ha formalizado su proceso de desarrollo y lo utiliza constantemente en sus proyectos. En un estudio que se realizó sobre los planes de desarrollo y los logs de ejecución de dos proyectos de esta empresa, se identificó que las actividades difieren, es decir, no hay una correspondencia simple entre las actividades que define el proceso, las que define el plan de desarrollo de los proyectos, y las que registran los logs de ejecución de los proyectos. Una herramienta que permita la generación automática de los planes de desarrollo, y que los cree directamente en la herramienta de gestión de proyectos que utiliza la empresa, reduciría significativamente este tipo de inconsistencias y permitiría al equipo de desarrollo utilizar el proceso como base de conocimiento; ya que las tareas a ejecutar estarían efectivamente documentadas en el proceso de la empresa.

1.2 Objetivos de la memoria

Este trabajo de memoria tiene como objetivo general el definir, diseñar e implementar una herramienta capaz de generar, en forma automática, planes de desarrollo para proyectos de software a partir del proceso de desarrollo de una empresa y de la información de los recursos asignados al proyecto.

Para alcanzar este objetivo general, se definen los siguientes objetivos específicos:

- Analizar soluciones y herramientas existentes para la creación y gestión de planes de desarrollo.
- Definir e implementar una solución que resuelva el problema.
- Validar la solución construida sobre un proceso de desarrollo de una empresa.

1.3 Plan de trabajo

El plan de trabajo seguido en la realización de la memoria involucró las siguientes actividades:

- Estudiar y analizar la funcionalidad de las herramientas y soluciones existentes que permitan la creación y gestión de planes de desarrollo de proyectos de software.
- Definir una arquitectura candidata del sistema de generación automática a partir de un proceso de desarrollo de planes de desarrollo.
- Implementar una versión funcional del sistema de generación de planes de desarrollo.
- Utilizar la herramienta aplicándola a sets de datos reales obtenidos de la empresa Ki Teknology.
- Ajustar la arquitectura y la implementación del sistema para resolver fallas y/o necesidades adicionales detectadas.

1.4 Estructura del documento

Este informe de memoria se estructura de la siguiente manera. En el Capítulo 2 se expone el marco teórico, en donde se presenta todos los conceptos y herramientas claves para entender este trabajo de título. En el Capítulo 3 se explica, en primer lugar, los requisitos de la solución y en segundo lugar los casos de uso de éste. En el Capítulo 4 se exhibe la solución al problema de la generación automática de planes de desarrollo a partir de procesos formalizados. En este capítulo se detalla todos los aspectos funcionales que esta requiere. En el Capítulo 5 se expone la implementación de la solución propuesta en el Capítulo 4. En el Capítulo 6 se detalla el proceso de la validación de la solución y la metodología usada para su validación. Finalmente, en el Capítulo 7 se exponen las conclusiones y trabajos futuros de este trabajo de título.

2 Marco teórico

En este capítulo se presentan los conceptos claves para el entendimiento del presente trabajo de título. En primer lugar se define el concepto de proceso de desarrollo. Posteriormente se precisa el concepto de plan de desarrollo de un proyecto. Finalmente se describen las herramientas y tecnologías necesarias para el desarrollo del proyecto de título.

2.1 Conceptos sobre procesos de desarrollo

En esta sección, primero se define lo que es un proceso de desarrollo, luego se introduce los lenguajes utilizados para formalizar un proceso de desarrollo. Finalmente se presenta la estructura utilizada para formalizar un proceso de desarrollo.

2.1.1 Definición de proceso de desarrollo

Un proceso de desarrollo de software es un conjunto parcialmente ordenado de actividades, los roles de las personas que las realizan y los artefactos que estos producen y utilizan al realizarlas, con el fin de crear y mantener un conjunto de entregables solicitados por un cliente [1]. Estas actividades, que se describen en la Sección 2.1.3, pueden construir aplicaciones desde cero. Sin embargo, generalmente el desarrollo de software se hace extendiendo y modificando desarrollos anteriores.

En Ingeniería del Software, un modelo de proceso de desarrollo de software puede verse como una manera de dividir el trabajo en distintas actividades (o el ciclo de vida del producto en distintas fases) con la intención de lograr la mejor gestión y el mejor resultado para el proyecto. Estos modelos pueden incluir la definición previa de entregables específicos y otros artefactos que son creados y completados por el equipo para diseñar, codificar, probar y mantener el software en cuestión. Sin importar qué modelo de desarrollo se elija: cascada, mediante prototipos, evolutivos, en espiral, en componentes, etc., cada uno de ellos está codificado en procesos de software.

2.1.2 Formalización de procesos de desarrollo

Existen diversos lenguajes que permiten formalizar procesos de desarrollo. En este trabajo se utilizó en particular uno de ellos, Unified Method Architecture (UMA). UMA es un lenguaje desarrollado por International Business Machines (IBM) el cual evolucionó del lenguaje de modelación, Software Process Engineering Meta-Model (SPEM) v1.1, el cual además integra conceptos de IBM Rational Unified Process, IBM Global Services y IBM Rational Summit Ascendant [7]. Además, a partir de UMA, nace la nueva especificación de SPEM 2.0 [7] [8].

La elección de utilizar el lenguaje de formalización UMA para este tema de memoria reside en dos puntos centrales. Uno, es que la herramienta de modelación de procesos, Eclipse Process Framework Composer, lo utiliza para formalizar procesos en él. El otro, es que los proyectos de investigación del proyecto GEMS ya están formalizados con éste.

2.1.3 Componentes de un proceso

Un proceso de desarrollo está conformado por un conjunto de actividades. Existen diversos tipos de actividades: fases, iteraciones, tareas e hitos.

- Fase: Cada uno de los distintos estados distintivos del proceso de desarrollo. Generalmente está formado por el resto de los componentes del proceso.
- Actividad: Un conjunto de tareas para lograr un propósito.
- Iteración: Es un conjunto de fases, actividades, tareas, hitos e inclusive más iteraciones que se repiten durante el proceso de desarrollo.
- Tarea: Describe las tareas que se tienen que hacer para lograr una actividad.
- Hito: Un hito marca un acontecimiento dado en el transcurso del proceso.

2.1.4 Formalización del Work Breakdown Structure del proceso

A partir del alcance del proyecto de desarrollo, se genera un listado de actividades, iteraciones, tareas e hitos a realizar, a esto se le llama Work Breakdown Structure (WBS); podemos ver un ejemplo en la Figura 2. El WBS contiene todos los componentes mencionados en la Sección 2.1.3, y además, contiene las dependencias de precedencia entre ellas, que son necesarias para llevar a cabo el proyecto.

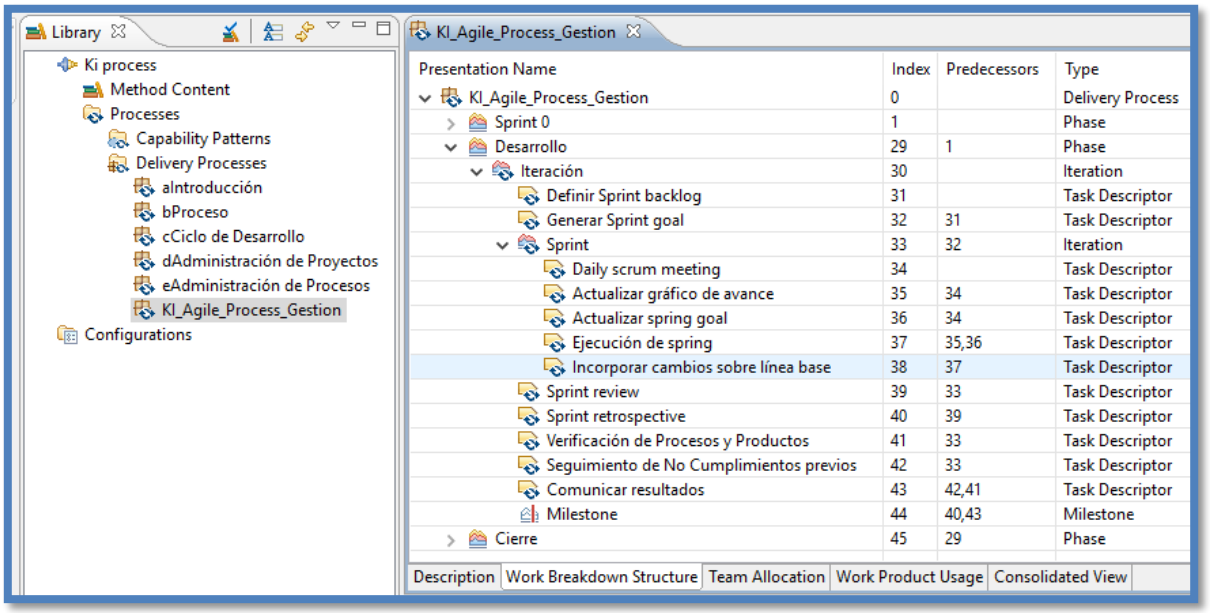


Figura 2: Work Breakdown Structure de un proceso de desarrollo

Además, existen dependencias lógicas entre las tareas del proyecto; generalmente estas dependencias se muestran mediante un diagrama de actividad. Un diagrama de actividad muestra gráficamente cómo las tareas componen una actividad del proceso; un ejemplo de un diagrama de actividad se puede ver en la Figura 3. Con toda esta información un jefe de proyecto es capaz de crear un plan de desarrollo manual o con algún software. Cabe destacar que en un plan de desarrollo el jefe de proyecto tiene que asignar los recursos disponibles al proyecto y organizar el calendario de las tareas [9].

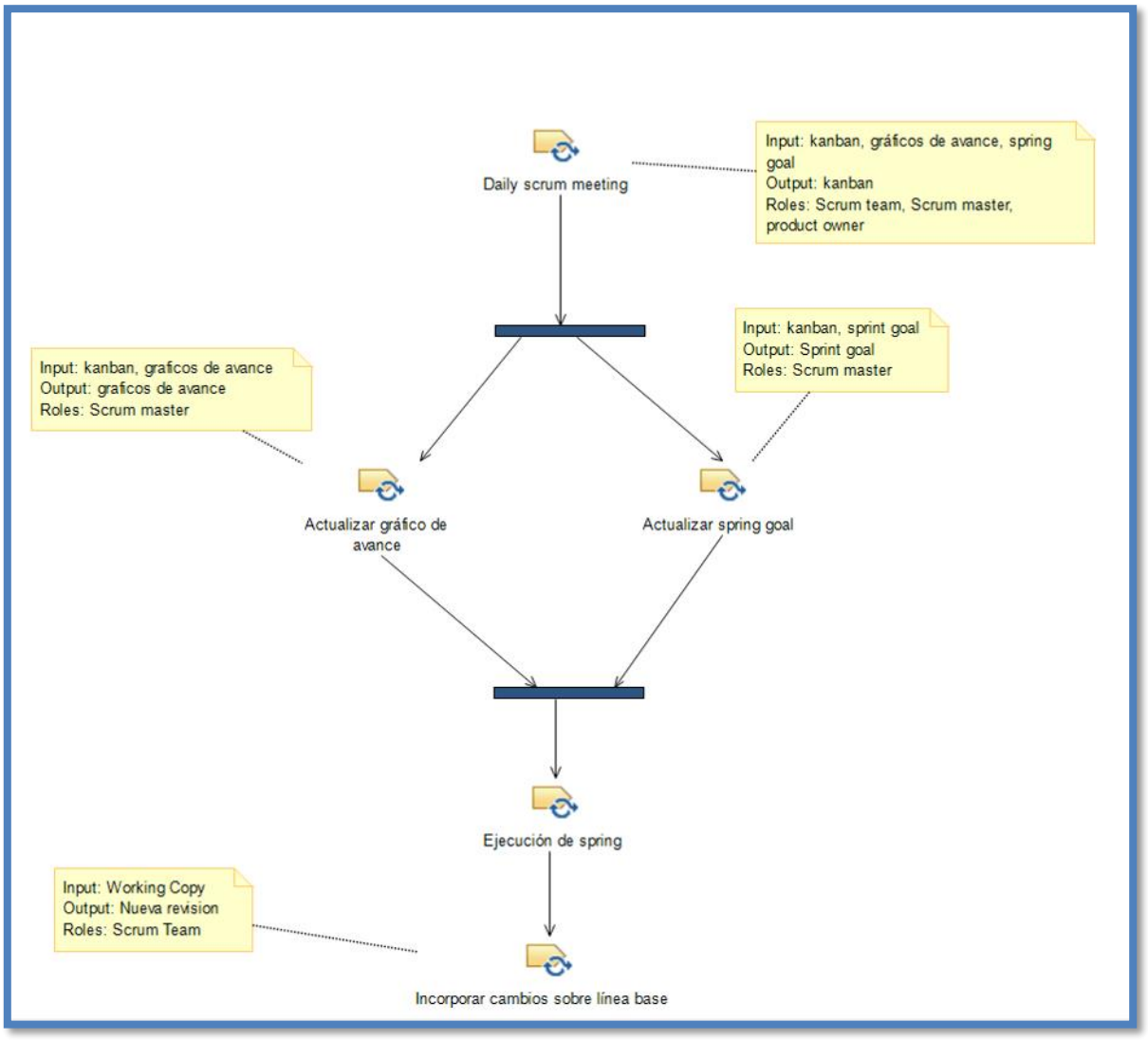


Figura 3: Diagrama de actividad de una actividad dentro de un proceso

Hay que destacar que un diagrama de actividades es más expresivo que un WBS. Este mayor nivel de expresividad lo logra el diagrama de actividades porque puede expresar condiciones y ciclos en el flujo de la actividad. Es decir, durante el transcurso de la actividad puede suceder que a partir de un resultado se tome una decisión y el flujo de la actividad varíe según la decisión tomada. Un ejemplo de esto se puede ver en la Figura 4 en donde se toma la decisión de si ha finalizado el sprint o no.

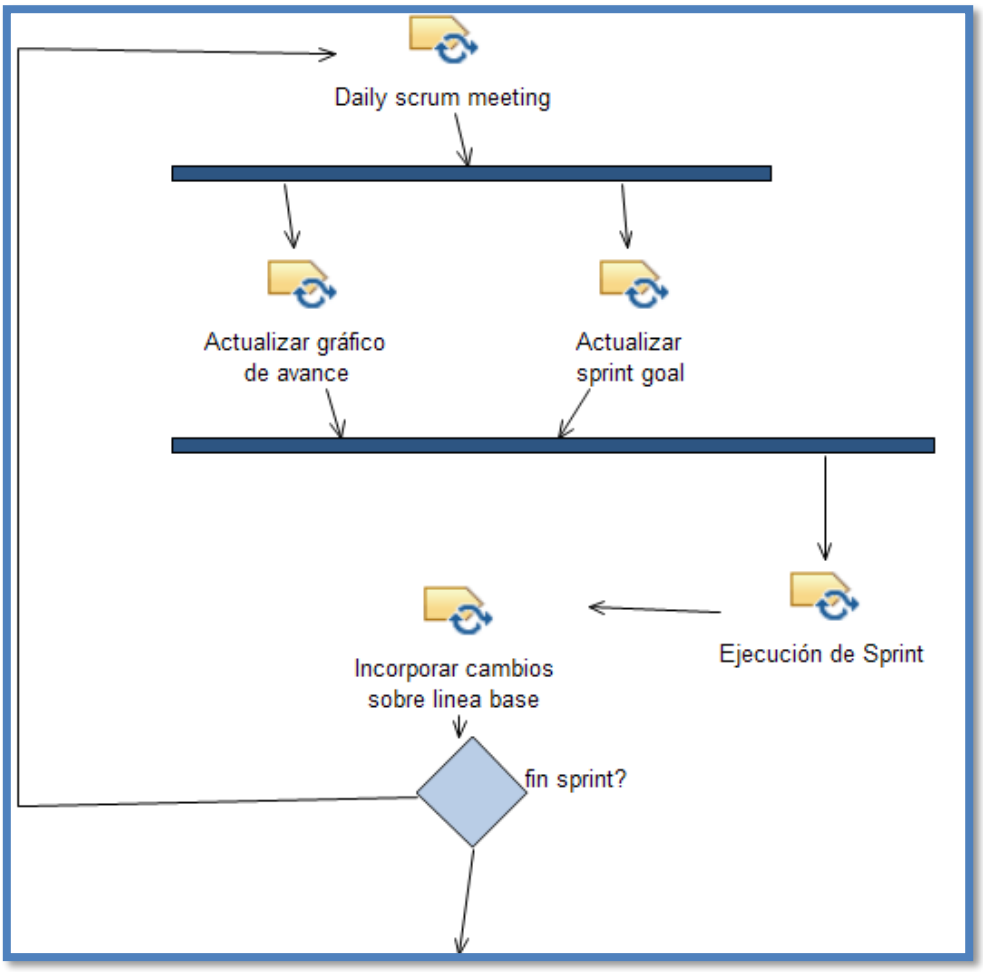


Figura 4: Diagrama de actividad de una actividad que contiene una condición

En la siguiente tabla podemos ver el nivel de expresividad del WBS y el diagrama de actividad.

	Diagrama de actividad	WBS
Secuencialidad	Sí	Sí
Paralelismo	Sí	Sí
Condicionalidad	Sí	No
Ciclo	Sí	Parcialmente con una iteración

Tabla 1: Tabla comparativa del nivel de expresividad del WBS y el diagrama de actividad

A pesar de ello, existe una directa relación entre los distintos componentes de un WBS y los diagramas de actividades que pueda contener el proceso de un proyecto.

Finalmente, a pesar de todas las características y facilidades que entrega un WBS, este carece de información esencial para generar un plan de desarrollo. En particular, el WBS no permite registrar la asignación de una actividad o tarea a un miembro del equipo ni el tiempo que tomará ejecutar cada tarea en un proyecto, aspectos esenciales en un plan de desarrollo.

2.2 Plan de desarrollo de un proyecto de software

En el inicio de un proyecto de software se define el o los objetivos de él, es decir cuál es el propósito de su realización, si crear un producto, un servicio o se busca algún resultado en particular. A partir de lo anterior se hace un análisis del alcance del proyecto y se obtiene todo el trabajo que se necesita realizar para cumplir con todas las características y funciones del producto que se desea realizar [10]. Con la informacion anterior se obtiene el WBS como se menciona en 2.1.4, ahora bien para obtener el plan de desarrollo para un proyecto de software, el jefe de proyecto debe asignar a cada actividad dentro del WBS su duracion y a quién está asignada para su realizacion. Finalmente se observa el resultado en la Figura 5, que muestra un plan de desarrollo a través de una carta Gantt.

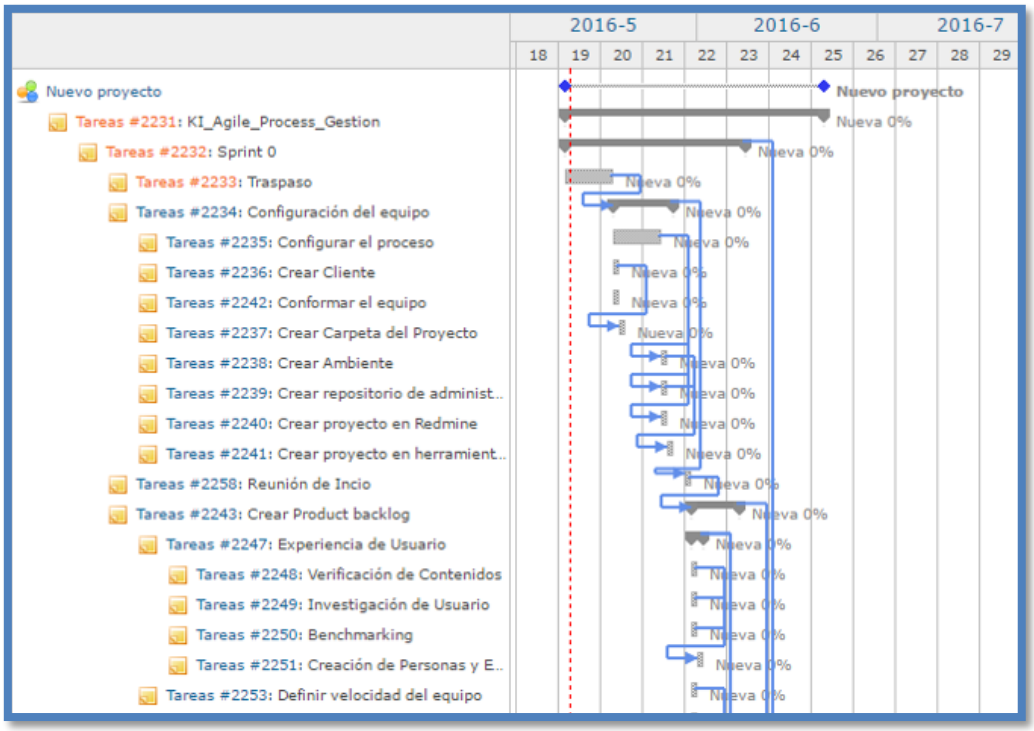


Figura 5: Vista de la carta Gantt de un plan de desarrollo en Redmine

2.3 Herramientas utilizadas

En la esta sección se describen las herramientas de software que se usaron durante el proyecto de memoria.

2.3.1 Eclipse Process Framework Composer

Eclipse Process Framework Composer (EPFC) es un software desarrollado por la fundación Eclipse y tiene dos objetivos principales. El primero es brindar un ambiente extensible y un conjunto de herramientas necesarias para la formalización de procesos de ingeniería. El segundo es proporcionar procesos extensibles para el desarrollo de software [7]. La herramienta EPFC utiliza para modelar su información el lenguaje UMA, que proviene del lenguaje SPEM que fue explicado en la Sección 2.1.2

EPFC utiliza la siguiente estructura para el modelamiento de procesos. En primer lugar, se crea un Method Library; esta contiene uno o más Method Plugins y las configuraciones de él. El Method Plugin se subdivide en dos secciones. Una es el Method Content; este define toda la información acerca de los artefactos, roles, tareas y guías que conforman los bloques de información del proceso en un WBS. La otra sección contiene dos tipos de procesos: Delivery Process y Capability Patterns. Se observa en la Figura 5 cómo es la estructura de un Method Library.

Los Delivery Process organizan de manera parcialmente ordenada el contenido del Method Content para un proceso en específico. En él se agrupan todos los componentes descritos en la Sección 2.1.3 y forman el WBS [11]. Un ejemplo de cómo se visualiza el WBS en EPFC es como se muestra en la Figura 2. Además, en esta herramienta, las tareas, dependiendo de su complejidad, pueden tener diagramas de actividad para lograr un mayor nivel de definición.

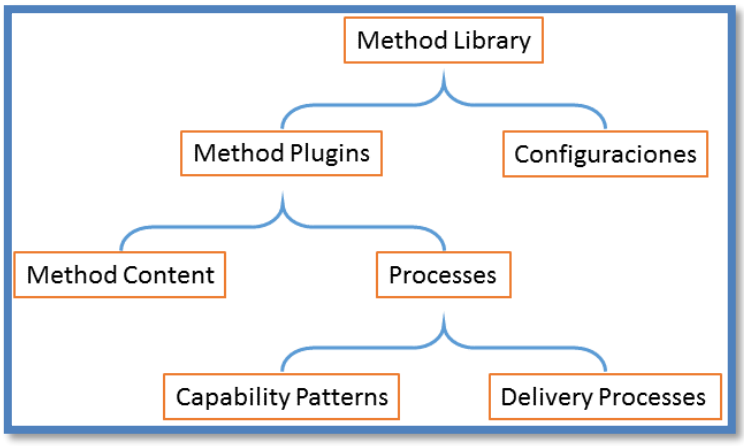


Figura 6: Esquema de información de EPFC

2.3.2 Redmine

Redmine es una plataforma open source. Esta plataforma web tiene como fin gestionar proyectos que incluyen: un sistema de seguimiento de incidentes (Bug Tracker), calendario de actividades, cartas Gantt y calendario, wiki, foro, sistema de integración de nuevas funcionalidades mediante plugins, etc. [12].

Esta herramienta favorece el control en la gestión de proyectos de desarrollos, a través de un plan de desarrollo que se crea dentro de la plataforma. Un ejemplo de ello se muestra en la Figura 5 donde se presenta una carta Gantt de un proyecto, es decir el WBS del plan de desarrollo del proyecto. En esta carta Gantt se observan las tareas que posee un proyecto, las duraciones y las precedencias entre ellas. Su primera versión fue lanzada en junio del 2006. Redmine está construido sobre el framework Ruby on Rails programado en el lenguaje Ruby y utiliza una base de datos MySQL.

3 Conceptualización de la solución

En este capítulo se presenta la conceptualización de la solución en términos de sus requisitos funcionales. En la Sección 3.1 se introducen las herramientas disponibles en el mercado. En la Sección 3.2 se detallan los requisitos para la formalización de una solución. Finalmente se exponen los casos de uso del sistema.

3.1 Selección de las herramientas

Existe un sinnúmero de herramientas para formalizar procesos de desarrollo (p.ej. EPFC, IBM Rational Method Composer) y para generar planes de desarrollo (p.ej. Redmine, OpenProject, Microsoft Project). Debido a esto, se hizo un estudio sobre las herramientas más populares, en este caso EPFC, Redmine (RM), OpenProject (OP) y Microsoft Project (MSP). Finalmente, se escogieron para la realización de este tema de memoria EPFC y Redmine.

Las herramientas escogidas para formalizar procesos (EPFC) y para generar planes de trabajos (Redmine) reside en dos puntos centrales. Primero ambas herramientas son open source y segundo ambas herramientas ya son utilizadas en Ki Teknology, que es la PyME de software chilena que proporcionó un proceso de desarrollo formalizado.

Cabe destacar que Redmine es una plataforma web ampliamente utilizada mundialmente en la industria de desarrollo de software [13]. Dentro de las empresas más destacables están Lenovo [14], Ruby y Lighttpd [15].

3.2 Análisis de requisitos funcionales

A continuación, se presentan los diferentes aspectos considerados dentro del levantamiento de requisitos asociados al diseño del sistema de software construido.

3.2.1 Flujo de la información en el proceso de formalización y creación de un plan de desarrollo

El proceso de formalización de un proceso y además la creación de un plan de desarrollo a partir de él, se constituye principalmente por tres actores: ingeniero de procesos, jefe de proyecto y equipo de desarrollo. Podemos ver en las Figuras 1 y 7 como estos interactúan.

En primer lugar, un ingeniero de procesos crea, genera y además mantiene el proceso formalizado de desarrollo de software de su empresa. En el caso de este trabajo el proceso se formaliza mediante la herramienta EPFC.

Paralelamente, existe el jefe de proyecto, que al comienzo de cada proyecto y a partir de los procesos formalizados, genera un plan de desarrollo. Este plan, dentro del margen de este proyecto, es formalizado en la plataforma de gestión de proyectos Redmine. En ella se crean todas las tareas que se encuentran en el proceso y se les asigna manualmente una duración y un miembro del equipo como responsable de dicha tarea.

Finalmente es el equipo de desarrollo de la empresa el que utiliza y gestiona el plan de desarrollo. En la Figura 7, podemos ver el proceso completo de la creación y utilización de un plan de desarrollo a partir de un proceso formalizado.

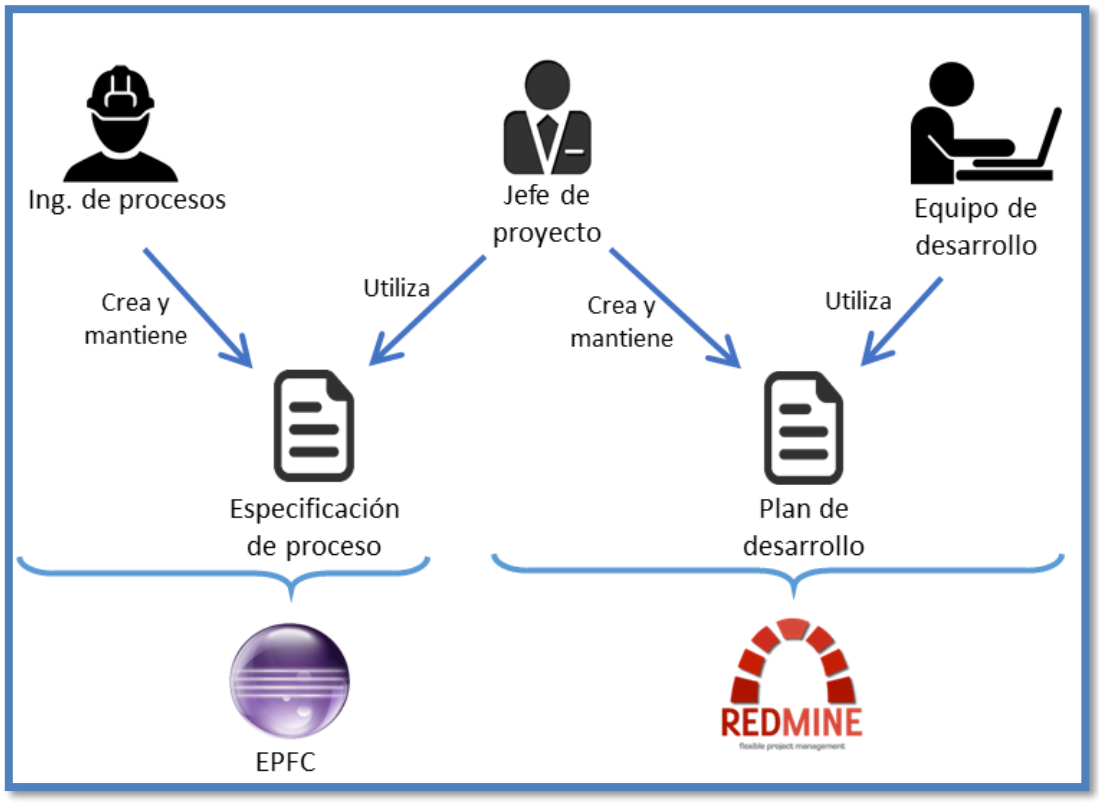


Figura 7: Diagrama de flujo de la información en el sistema

3.2.2 Requisitos de la solución

El objetivo de la solución es implementar un mecanismo que permita automatizar la creación de un plan de desarrollo desde un proceso de desarrollo formalizado. Es decir, lograr extraer un proceso de desarrollo de EPFC y generar con él un plan de desarrollo en Redmine.

En el caso particular del sistema que extraiga la información desde EPFC y la empaquete en algún artefacto de intercambio legible (p.ej. XML, JSON, CSV), es necesario tener instalado en una máquina EPFC v1.5.1.7 y en él tener modelado el proceso que se quiere utilizar como base para el plan de desarrollo. El proceso tiene que estar modelado en su totalidad y expresado fidedignamente en el Work Breakdown Structure (WBS). La importancia de tener un WBS fidedigno al proceso es que el sistema obtendrá la información del WBS únicamente y no de los diagramas de actividad que también contiene un proceso formalizado en EPFC. En caso de que no fuese así, debido al problema mencionando en la Sección 2.1.4, el artefacto de intercambio no contendría toda la información del proceso.

Ahora bien, como se mencionó en la Sección 2.1.4, el WBS y los diagramas de actividad tienen un grado de correlación por lo tanto, lo que realmente se requiere es que no existan actividades que contengan ciclos. Consecuentemente se busca que el proceso no contenga condiciones que puedan generar ciclos. Debido a esto, una actividad que contenga una condición que pueda genera un ciclo, debe adaptarse a una iteración, lo que conlleva la pérdida de cierto grado de información de la actividad.

A modo de ejemplificar el proceso de adaptación de una actividad que genera un ciclo, se usará una actividad del proceso de Ki Teknology. Se observa en la Figura 8 el diagrama de actividad dentro de una actividad que contiene un ciclo. El recuadro rojo sobre la figura marca dónde se ubica el ciclo dentro de la actividad. Además, se observa en la Figura 9 cómo es el proceso no alterado en un WBS.

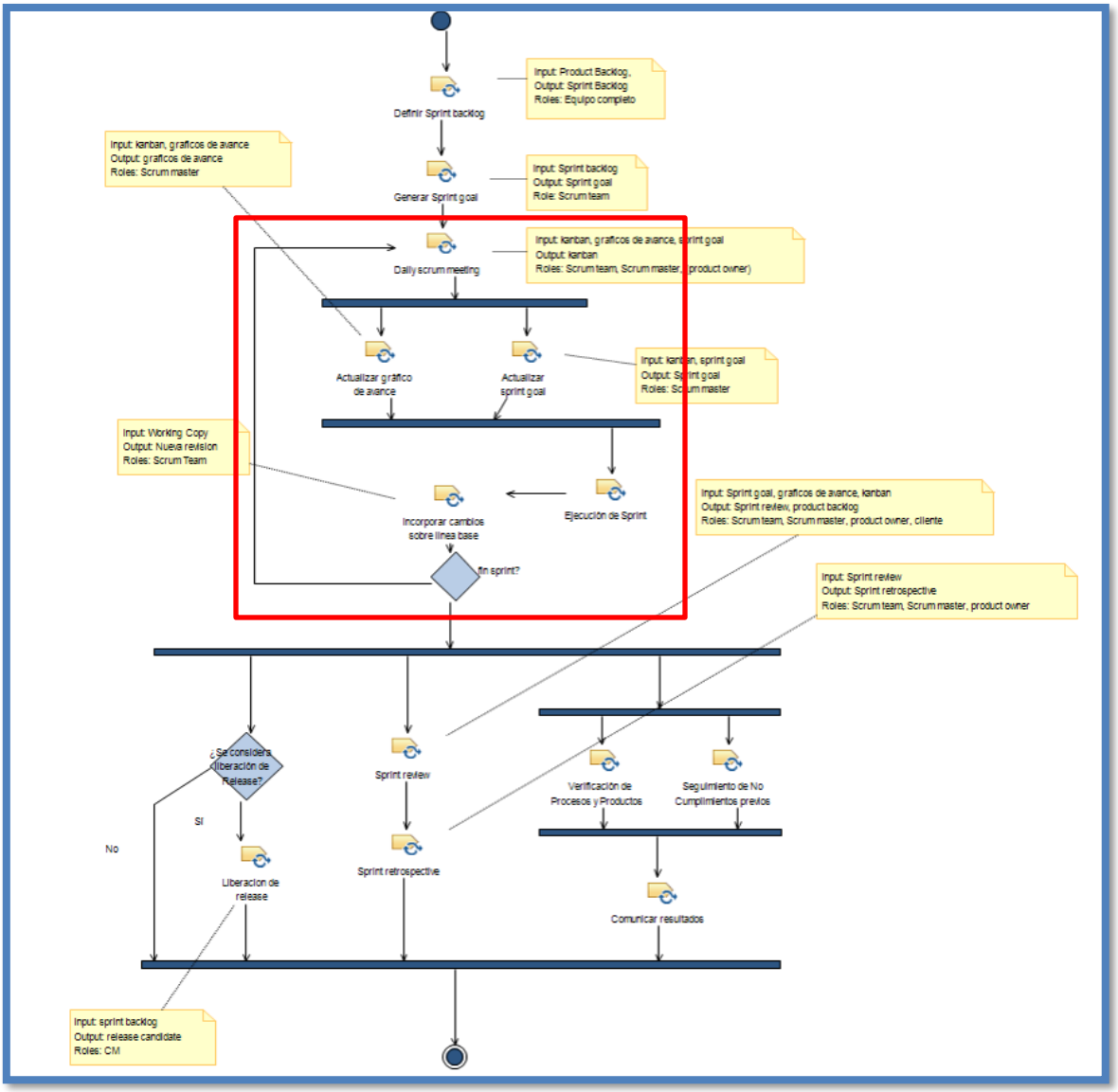


Figura 8: Diagrama de actividad que contiene una condición que genera un ciclo

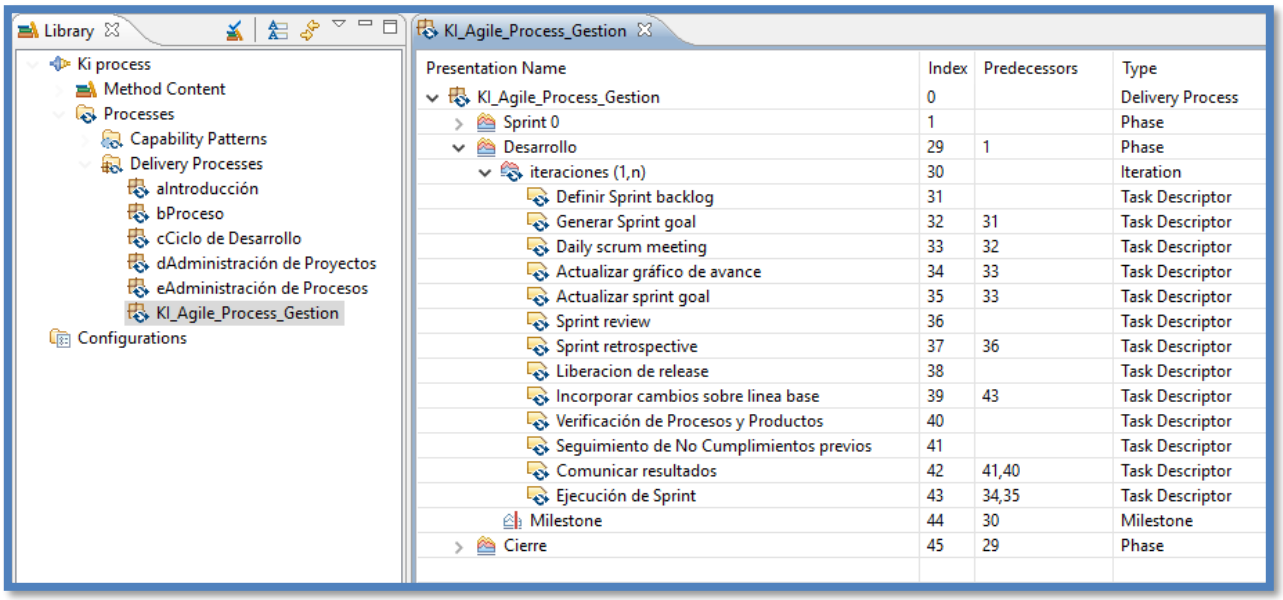


Figura 9: WBS del proceso no adaptado

A pesar de que esta actividad posee un ciclo, se puede adaptar para que no lo contenga. Para lograr esto se sustituye el ciclo por una iteración y todas las actividades que existían en el ciclo se situaron dentro de la iteración. Se observa en la Figura 10 el reemplazo de los componentes del ciclo por una iteración.

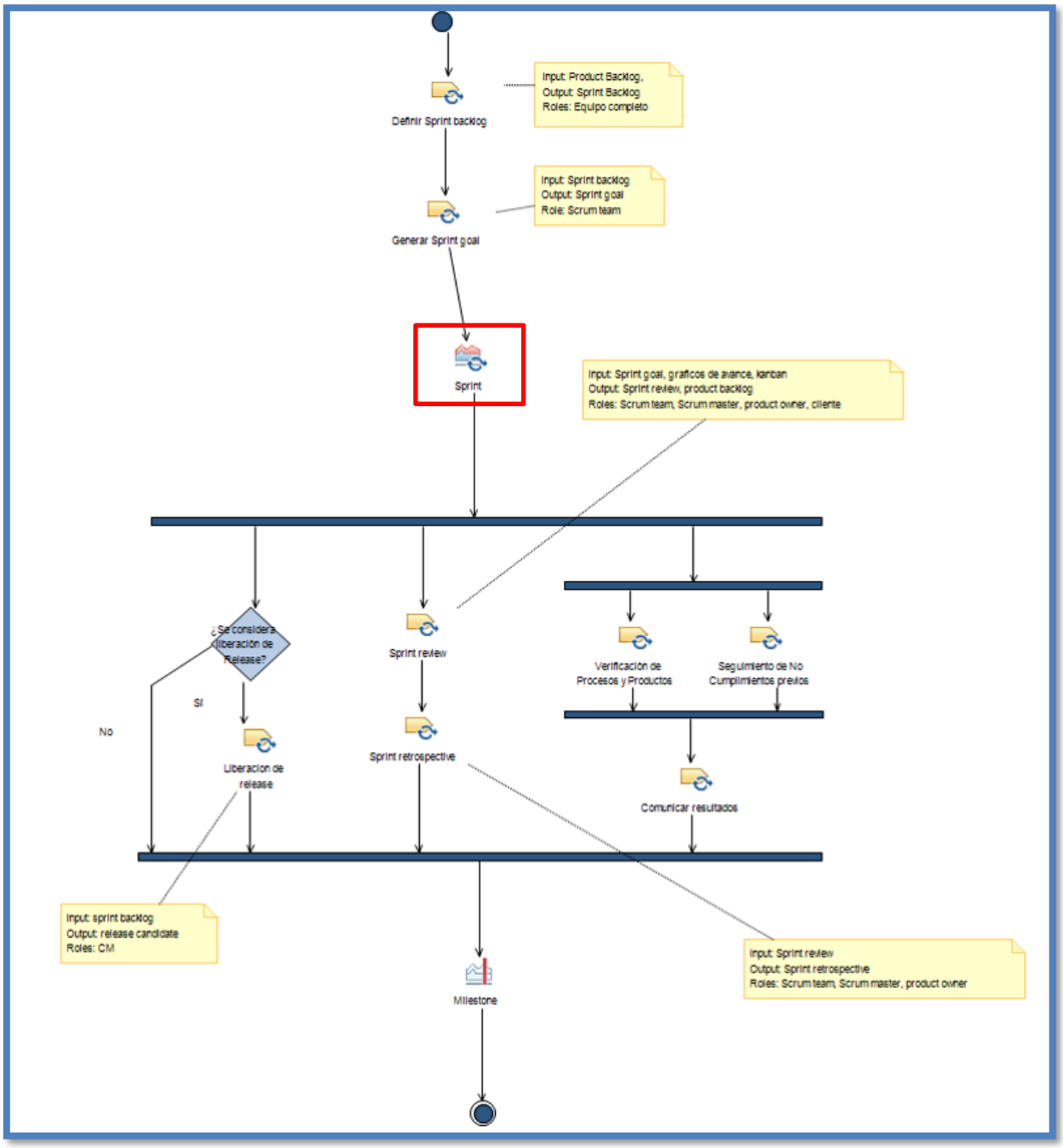


Figura 10: Diagrama de actividad adaptado

Finalmente, se pueden observar en la Figura 11, las actividades que anteriormente estaban afuera de la iteración, formando el ciclo, que en el modelo conforman una iteración dentro de la actividad.

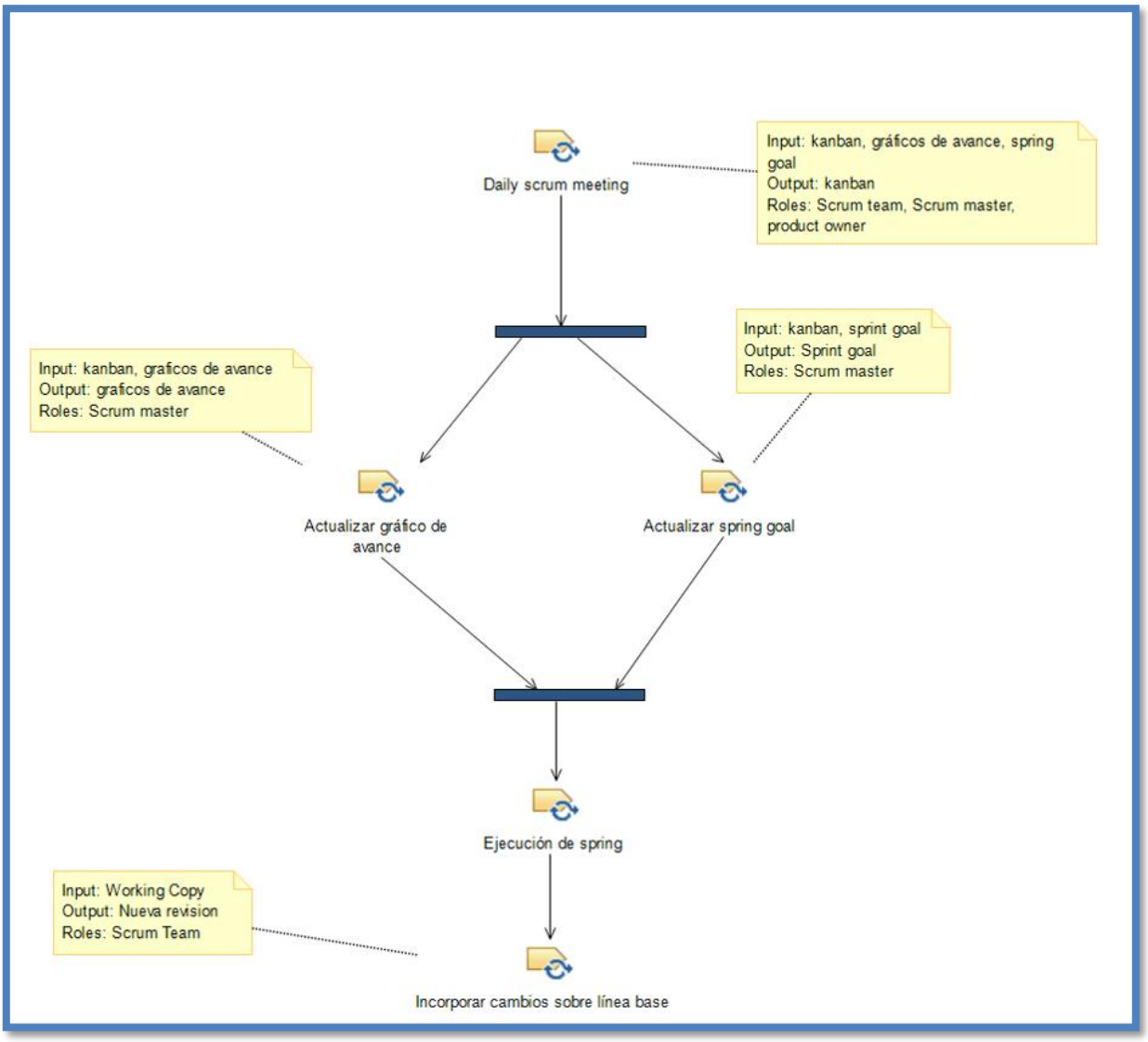


Figura 11: Componentes de la iteración de la actividad adaptada

Se observa en la Figura 2 como es el WBS del proceso adaptado.

En el caso particular de la solución para Redmine, este deberá correr dentro de la plataforma Redmine v3.1.1, la cual a su vez necesita del lenguaje Ruby v2.0, el framework Ruby On Rails 4.2.4 y además hace uso de la base de datos MySQL v5.5. El sistema debe ser capaz de consumir la información que genere el sistema que se integra con EPFC.

3.3 Casos de uso del sistema

En esta sección se exponen los casos de uso del sistema.

3.3.1 Sistema

La Figura 12 muestra el diagrama UML de los casos de uso del sistema.

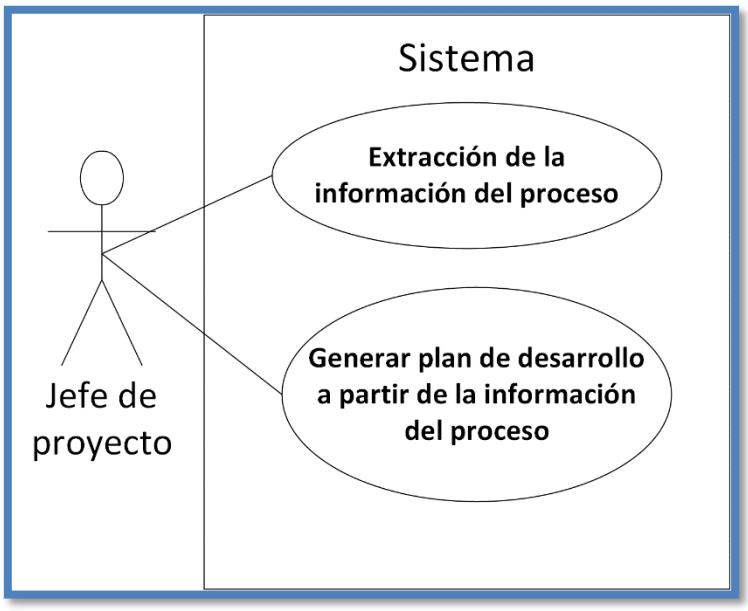


Figura 12: Casos de uso del sistema

3.3.1.1 Extracción de la información del proceso (CU1)

En este caso de uso, el jefe de proyecto exporta toda la información del proceso formalizado en EPFC a un formato legible.

Nombre	Extracción de la información del proceso
Actores	Jefe de proyecto
Sinopsis	Este caso de uso comienza cuando el jefe de proyecto abre la aplicación EPFC. A continuación, ejecuta el sistema dentro de EPFC y selecciona el proceso que se le quiere extraer la información. Al ejecutarlo, este exportara la información a un artefacto de intercambio en un formato legible.
Curso Típico de Eventos	
	1. El jefe de proyecto abre EPFC. 2. El jefe de proyecto abre el sistema y selecciona el proceso que se quiere extraer. 3. El sistema analiza el proceso y determinara si existen iteraciones. 4. El jefe de proyecto ingresa el nombre del proyecto al cual se exportarán todas las actividades del proceso. Si no hay iteraciones no le pedirá ningún otro dato al usuario. 5. El sistema generará un artefacto de intercambio con la información extraída y lo guardará.
Extensiones	
	4a. Si existen iteraciones dentro del proceso: 1. Se despliegan todas las iteraciones que existen en el proceso. 2. El jefe de proyecto ingresa la cantidad de veces que quiere que se repita cada iteración del proceso. 3. Retoma en 5.

Tabla 2: Caso de uso CU1

3.3.1.2 Generar plan de desarrollo a partir de la información del proceso (CU2)

En este caso de uso, el jefe de proyecto importa toda la información del proceso desde un artefacto de intercambio, con el fin de generar un plan de desarrollo en la plataforma destino.

Nombre	Generar plan de desarrollo a partir de la información del proceso
Actores	Jefe de proyecto
Sinopsis	Este caso de uso comienza cuando el jefe de proyectos abre la plataforma Redmine. Ahí el jefe de proyecto selecciona el plugin, en donde se seleccionara el archivo que contiene la información, en un formato legible para el sistema, que se desea importar a Redmine.
Curso Típico de Eventos	
	1. El jefe de proyecto abre Redmine en un explorador web (p.ej. Chrome, IE). 2. El jefe de proyecto, dentro de la plataforma, selecciona importar un proyecto. 3. El jefe de proyecto selecciona el artefacto de intercambio generado en el caso de uso 1. 4. El sistema genera automáticamente el plan de desarrollo a partir del archivo que contiene la información.

Tabla 3: Caso de uso CU2

4 Diseño de la solución

El sistema construido consiste en dos plugins y además en la definición de un artefacto de intercambio, tal como se ilustra en la Figura 13.

El primer plugin llamado `XmlGen`, transforma y exporta las actividades modeladas en EPFC a un artefacto de intercambio. El segundo plugin llamado `XmlImport` del sistema, consume el artefacto de intercambio y traspasa la información necesaria para que Redmine genere el plan de desarrollo en él. Finalmente, con el plan de desarrollo generado, es el jefe de proyecto el que debe asignar la duración de la actividad y asignar dentro del equipo de desarrollo a quién pertenece cada tarea.

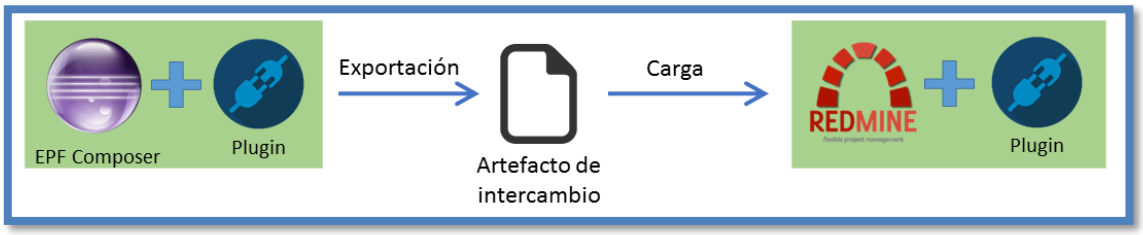


Figura 13: Esquema de la de solución

Con el objetivo de describir la estructura y funcionamiento del sistema desarrollado en relación a los plugins `XmlGen` y `XmlImport`, se presentarán diagramas en base al Lenguaje de Modelamiento Unificado versión 2.0 (UML 2.0) [16]. El diseño de la solución se estructura en términos de vistas de arquitectura basadas en el catálogo de vistas propuestas en [17]. La Sección 4.1 presenta la vista funcional, en la Sección 4.2 la vista de información y finalmente en la Sección 4.3 la vista de deployment.

4.1 Vista funcional

El sistema consta de tres componentes centrales, un plugin que interactúa con EPFC llamado `XmlGen`, otro que interactúa con Redmine llamado `XmlImport` y finalmente un artefacto de intercambio entre ellos. El plugin `XmlGen` tiene como objetivo extraer y analizar la información del proceso formalizado en EPFC y transformarlo para luego grabarlo a un archivo XML. Por otra parte, el plugin `XmlImport` tiene como objetivo leer y analizar la información del archivo XML e importarla a la plataforma Redmine, creando en ella un proyecto y un plan de desarrollo.

A continuación, se muestra un diagrama de la vista funcional. En ella se expone cómo interactúan los componentes que conforman la solución del sistema de automatización de la generación de un plan de desarrollo.

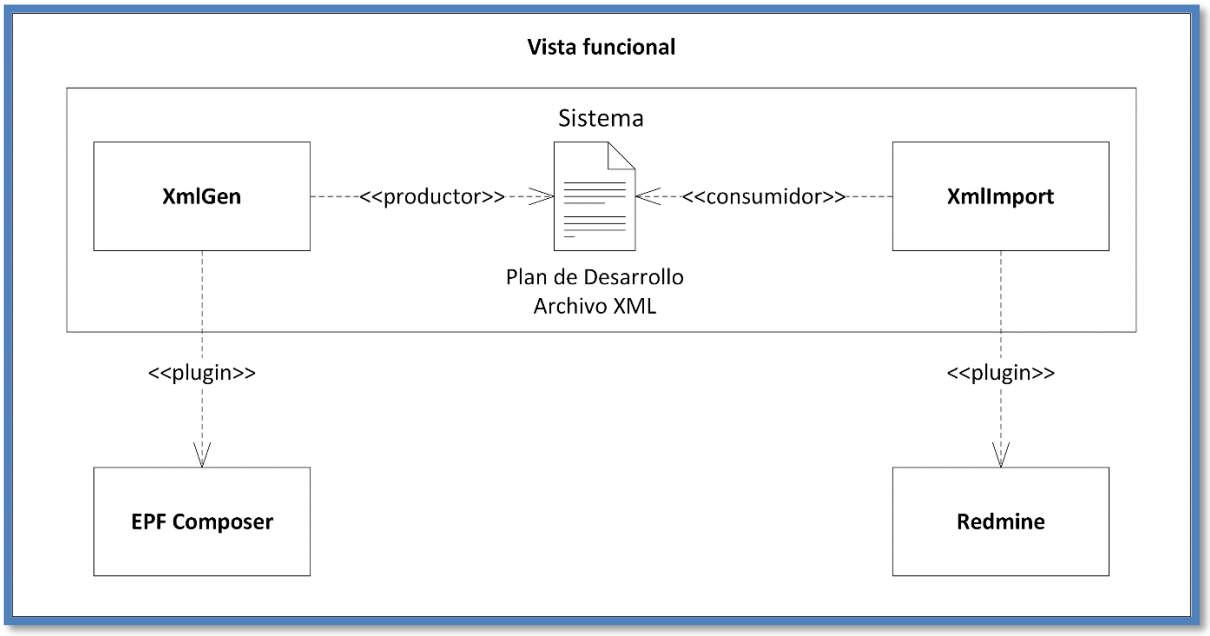


Figura 14: Diagrama de la vista funcional global del sistema

A continuación, se detallará con mayor profundidad la composición funcional de `XmlGen` y `XmlImport`.

`XmlGen` está compuesto por dos clases `XmlGenPlugin` y `Wizard`. La primera clase es la que maneja toda la lógica de consumir, transformar y exportar el proceso a un artefacto de intercambio. Éste utiliza primordialmente la librería `epf.uma` para poder consumir directamente el proceso de EPFC y utiliza la librería `dom4j` para guardar la información a un archivo XML. La segunda clase `Wizard`, es la que maneja todo el aspecto visual del plugin. Ésta hereda todas sus funciones de la clase `Wizard` provista por `eclipse.jface.wizard`. Se puede ver en la Figura 15 la vista funcional de `XmlGen` y sus dependencias.

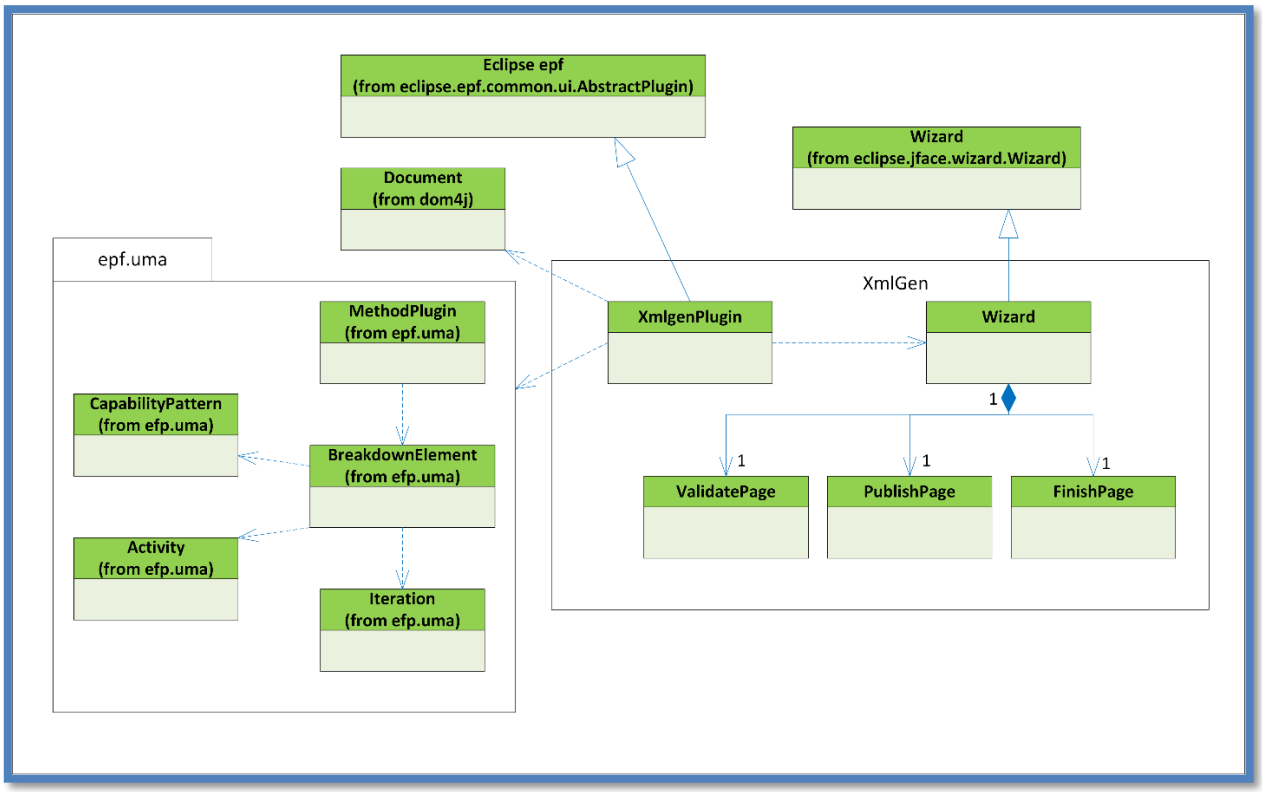


Figura 15: Diagrama de clases de XmlGen

Finalmente, el plugin `XmlImport` está compuesto por dos clases, `XmlImportController` y `XmlTask`. La primera clase maneja toda la lógica para la lectura, análisis e importación de la información del archivo XML. Para ello esta clase hace uso de las librerías `REXML` y de la API de Redmine. Por otro lado, está la clase `XmlTask`, que representa a los Tasks definidos en el artefacto de intercambio. Se puede ver en la Figura 16 la vista funcional de `XmlImport` y sus dependencias.

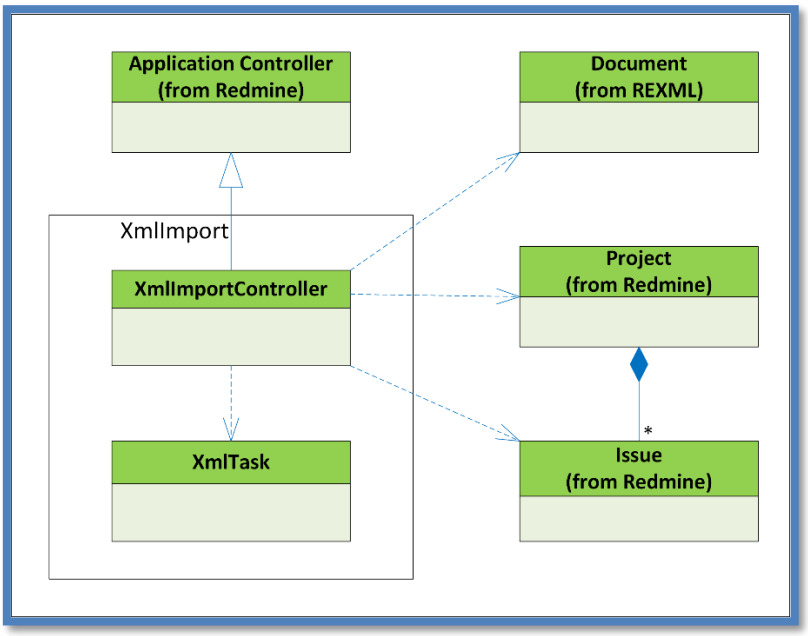


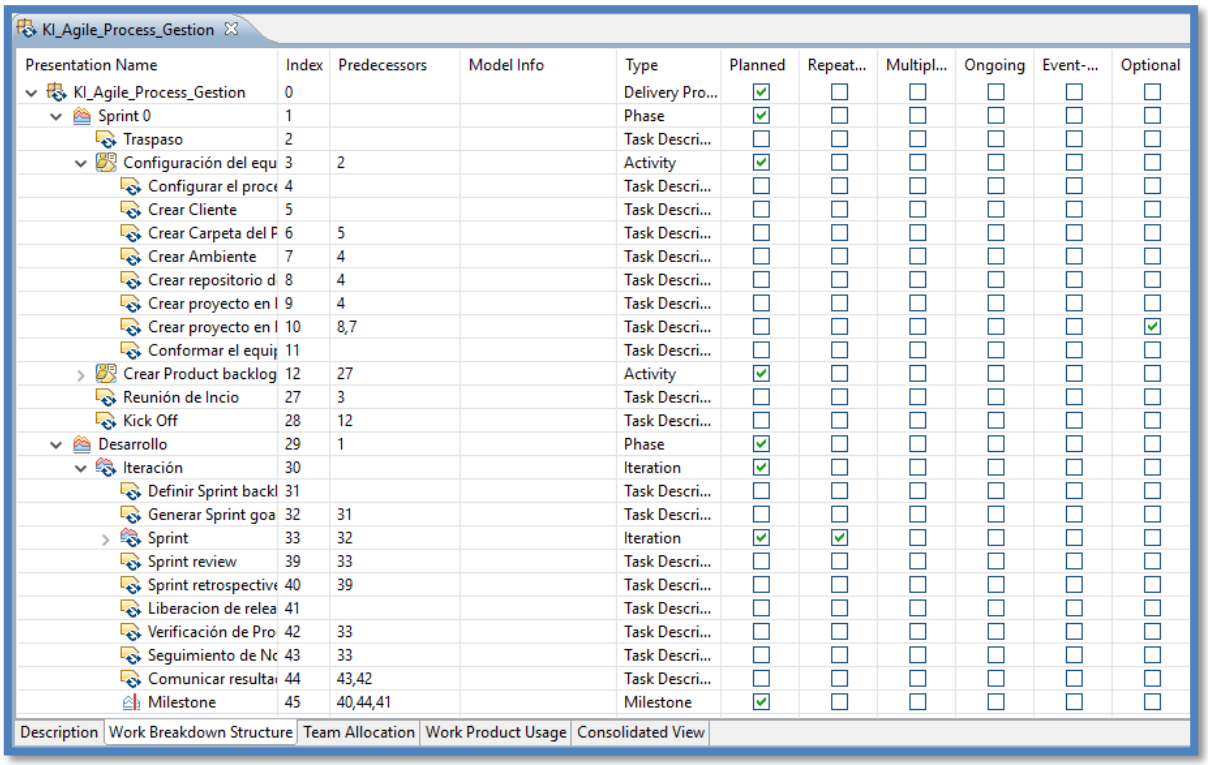
Figura 16: Diagrama de clases de XmlImport

4.2 Vista de información

En esta sección se describe la estructura de la información dentro de cada componente de software usada por la solución.

4.2.1 Eclipse Process Framework Composer

Para este trabajo de memoria se utilizó el Work Breakdown Structure (WBS) de Eclipse Process Framework Composer (EPFC). Este lenguaje describe en forma de árbol, el proceso mediante sus distintos componentes mencionados en la Sección 2.1.3. Cabe destacar que el WBS guarda toda la información acerca de un proceso con excepción de la duración de éste y a qué miembro del equipo le corresponde ejecutarla, ya que esta información corresponde al de un plan de desarrollo y no al WBS de un proceso. Además de ello, todas las actividades modeladas en EPFC pueden contener la información de su antecesor. Se puede ver un ejemplo de WBS de un proceso en la Figura 17.



Presentation Name	Index	Predecessors	Model Info	Type	Planned	Repeat...	Multipl...	Ongoing	Event-...	Optional
KI_Agile_Process_Gestion	0			Delivery Pro...	☑	☐	☐	☐	☐	☐
Sprint 0	1			Phase	☑	☐	☐	☐	☐	☐
Traspaso	2			Task Descri...	☐	☐	☐	☐	☐	☐
Configuración del equ	3	2		Activity	☑	☐	☐	☐	☐	☐
Configurar el proce	4			Task Descri...	☐	☐	☐	☐	☐	☐
Crear Cliente	5			Task Descri...	☐	☐	☐	☐	☐	☐
Crear Carpeta del P	6	5		Task Descri...	☐	☐	☐	☐	☐	☐
Crear Ambiente	7	4		Task Descri...	☐	☐	☐	☐	☐	☐
Crear repositorio d	8	4		Task Descri...	☐	☐	☐	☐	☐	☐
Crear proyecto en I	9	4		Task Descri...	☐	☐	☐	☐	☐	☐
Crear proyecto en I	10	8,7		Task Descri...	☐	☐	☐	☐	☐	☑
Conformar el equi	11			Task Descri...	☐	☐	☐	☐	☐	☐
Crear Product backlog	12	27		Activity	☑	☐	☐	☐	☐	☐
Reunión de Incio	27	3		Task Descri...	☐	☐	☐	☐	☐	☐
Kick Off	28	12		Task Descri...	☐	☐	☐	☐	☐	☐
Desarrollo	29	1		Phase	☑	☐	☐	☐	☐	☐
Iteración	30			Iteration	☑	☐	☐	☐	☐	☐
Definir Sprint backl	31			Task Descri...	☐	☐	☐	☐	☐	☐
Generar Sprint goa	32	31		Task Descri...	☐	☐	☐	☐	☐	☐
Sprint	33	32		Iteration	☑	☑	☐	☐	☐	☐
Sprint review	39	33		Task Descri...	☐	☐	☐	☐	☐	☐
Sprint retrospective	40	39		Task Descri...	☐	☐	☐	☐	☐	☐
Liberacion de relea	41			Task Descri...	☐	☐	☐	☐	☐	☐
Verificación de Pro	42	33		Task Descri...	☐	☐	☐	☐	☐	☐
Seguimiento de Nc	43	33		Task Descri...	☐	☐	☐	☐	☐	☐
Comunicar resulta	44	43,42		Task Descri...	☐	☐	☐	☐	☐	☐
Milestone	45	40,44,41		Milestone	☑	☐	☐	☐	☐	☐

Figura 17: WBS de un proceso en EPFC

En la Figura 18 se muestra una vista simplificada de la estructura de datos del WBS de EPFC. En ella, podemos ver que existe, un `Method Library` el cual contiene `Method Plugins`. Un `Method Plugin` contiene `Method Content` y `Processes`, finalmente `Processes` contiene `Delivery Processes` que es donde se representa el proceso de desarrollo.

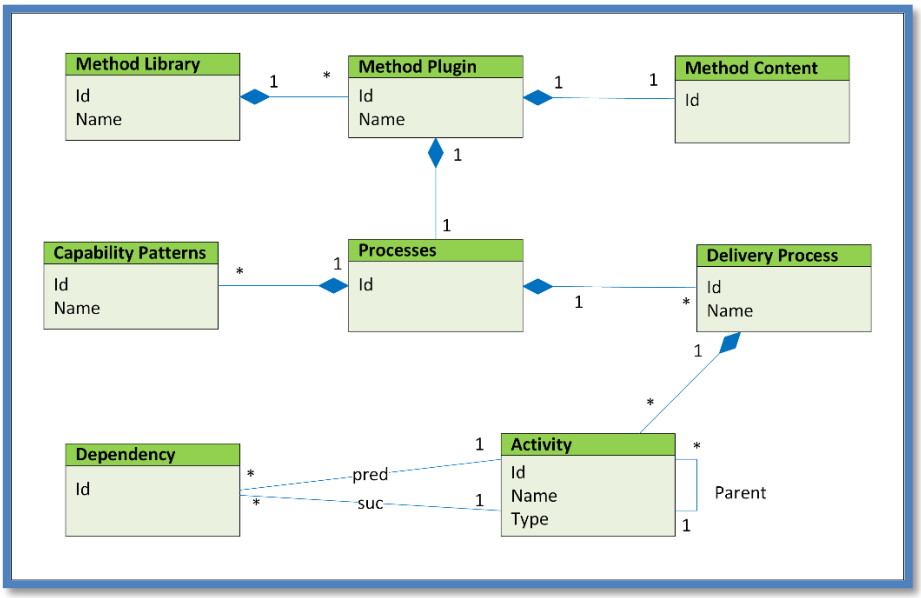


Figura 18: Modelo de datos del WBS en EPFC

4.2.2 Redmine

Redmine es la plataforma que se usó para alojar el plan de desarrollo que se genera a través del proceso de desarrollo modelado en EPFC. Ésta modela todos los componentes del proceso como tareas conservando la precedencia entre ellos. Por lo tanto, cada componente de un proceso: actividad, tarea, iteración, fase e hito, en Redmine son tareas a las cuales se les puede asignar una duración y quién es el encargado de realizarla. A continuación, se muestra un plan de desarrollo generado a partir del proceso de la figura anterior.

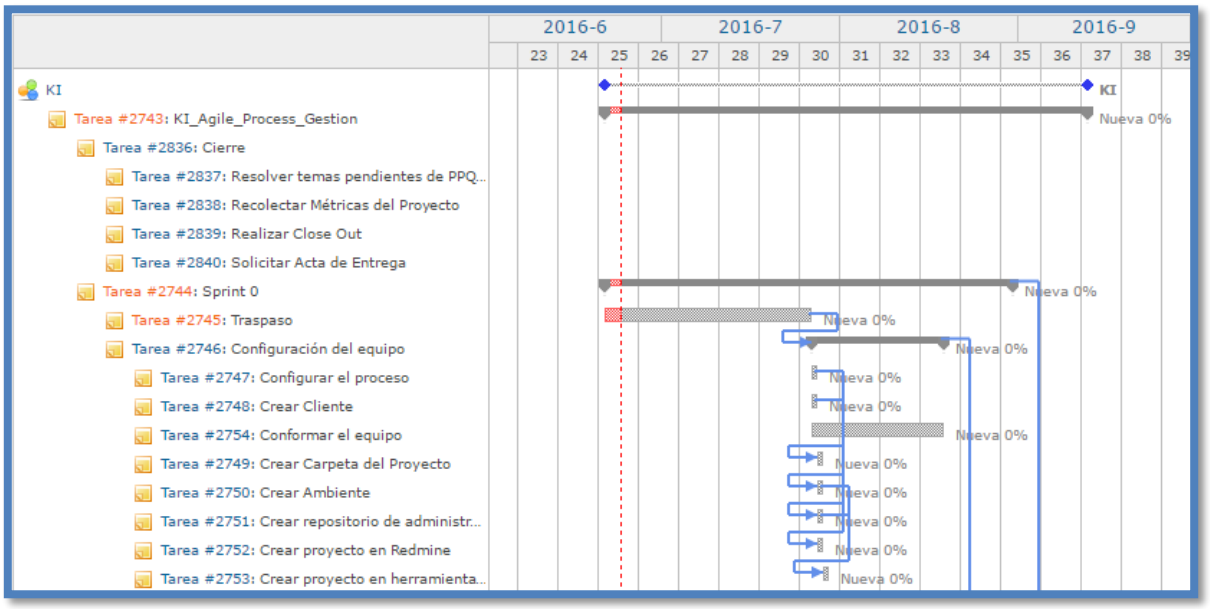


Figura 19: Vista de la carta Gantt de un plan de desarrollo

Una versión simplificada del modelo de datos que usa Redmine para guardar toda la información de un proyecto se muestra en la Figura 20. Se observa que un proyecto contiene miembros y tareas. Una tarea tiene asociada su precedencia y a usuarios en caso de que se asigne alguno a ella. Finalmente un usuario tiene un rol dentro de la plataforma.

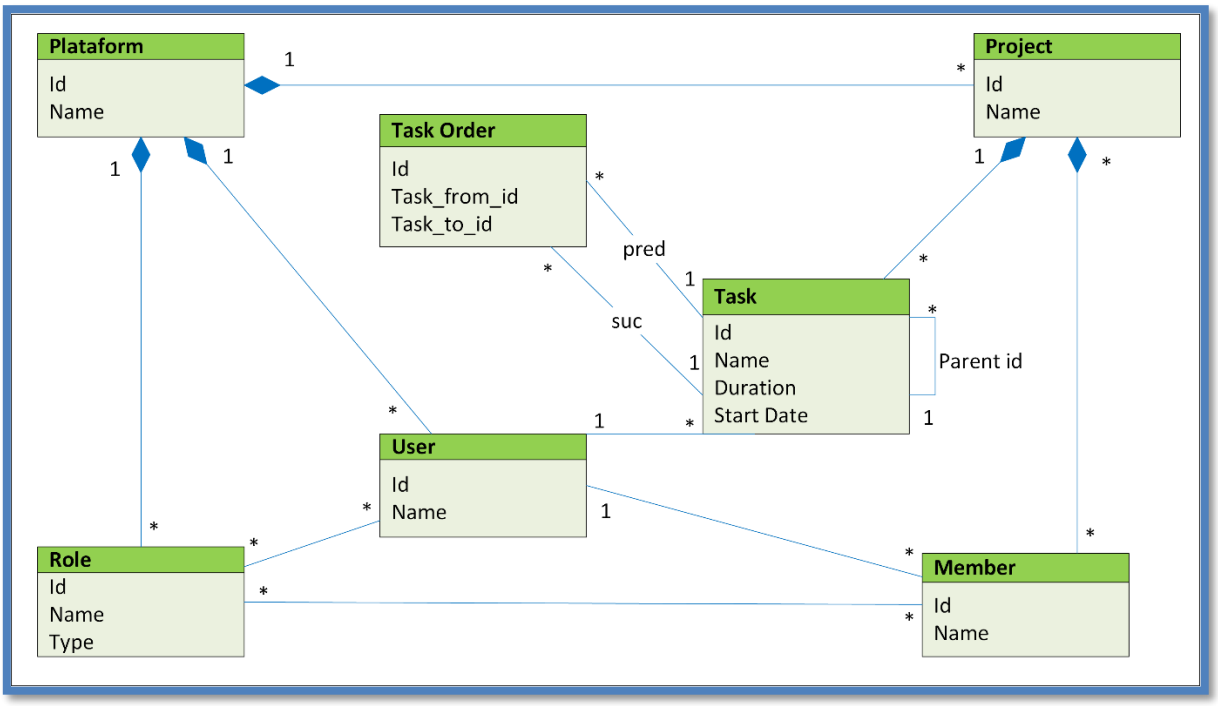


Figura 20: Diagrama del modelo de datos de Redmine

4.2.3 Artefacto de intercambio

El sistema se compone de dos plugins en dos plataformas distintas y debido a la naturaleza de la solución, se requiere que éstos intercambien la información del proceso de alguna manera. Para ello se creó un artefacto de intercambio entre los plugins, el que consiste de un archivo XML. Debido a lo anterior, el objetivo de generar un artefacto de intercambio, es que los plugins involucrados no tengan dependencias con la estructura de información generada dentro del artefacto de intercambio.

Este archivo XML contiene toda la información generada de `XmlGen` y debe ser capaz de ser consumido por `XmlImport`. Es por ello que se generó un esquema para el archivo XML con las siguientes características. En primer lugar, contiene la información del proyecto con los siguientes elementos: id, nombre y descripción. Luego un proyecto tiene tareas; este tiene los siguientes elementos: id, nombre, tipo, estado, prioridad, descripción, a quién sigue y quién es su padre. Cabe destacar que el archivo XML es un árbol de punteros al padre y además contiene punteros que indican a quién sigue, para lograr mantener la precedencia de las tareas. En la Figura 21, se observa el esquema del modelo de datos del archivo XML. Finalmente gracias al artefacto de intercambio que contiene toda la información, éste es consumible en un futuro por alguna otra herramienta en caso de que fuese necesario.

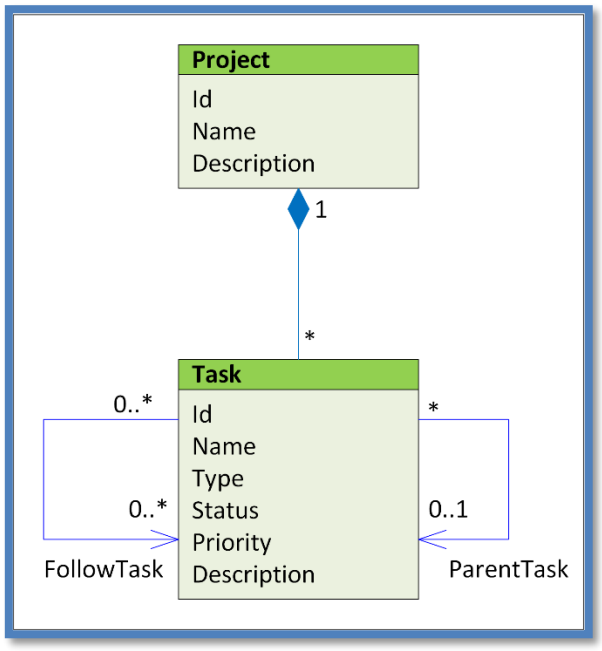


Figura 21: Diagrama del modelo de información del artefacto de intercambio

Se observa en la tabla 4, cómo son las equivalencias de las tablas de los modelos de datos de EPFC y Redmine y el modelo de información del artefacto de intercambio. A partir de estas equivalencias `XmlGen` genera la estructura en el archivo XML que contiene toda la información del proceso formalizado en EPFC. Es decir, transforma las Activities a Tasks, las Dependencies a FollowTasks y finalmente le pregunta al usuario el nombre del proyecto. De la misma manera `XmlImport` extrae la información del XML y hace la siguiente transformación, convierte de Tasks a Tasks, de FollowTasks a TaskOrders y de Project a Project.

EPFC	Artefacto de intercambio	Redmine
Activity	Task	Task
Dependency	FollowTask	Task Order
Input de usuario	Project	Project

Tabla 4: Equivalencia entre los modelos de datos de EPFC y Redmine y el modelo de información del artefacto de intercambio

A continuación, se muestra un ejemplo del esquema del archivo XML generado a partir del proceso de Ki Teknology.

```
<Project>
  <Name>KI</Name>
  <Description></Description>
  <Id>KI</Id>
  <Tasks>
    <Task>
      <Id>1</Id>
      <Name>KI_Agile_Process_Gestion</Name>
      <Type>0</Type>
      <ParentTask>0</ParentTask>
      <Status>status</Status>
      <Priority>priority</Priority>
      <Description></Description>
    </Task>
    <Task>
      <Id>2</Id>
      <Name>Sprint 0</Name>
      <Type>0</Type>
      <ParentTask>1</ParentTask>
      <Status>status</Status>
      <Priority>priority</Priority>
      <Description></Description>
    </Task>
    <Task>
      <Id>3</Id>
      <Name>Traspaso</Name>
      <Type>0</Type>
      <ParentTask>2</ParentTask>
      <Status>status</Status>
      <Priority>priority</Priority>
      <Description></Description>
    </Task>
    <Task>
      <Id>4</Id>
      <Name>Configuración del equipo</Name>
      <Type>0</Type>
      <ParentTask>2</ParentTask>
      <Status>status</Status>
      <Priority>priority</Priority>
      <Description></Description>
      <Follows>3</Follows>
    </Task>
     ……
  </Tasks>
</Project>
```

Tabla 5: Esquema de artefacto de intercambio

4.3 Vista de deployment

En la Figura 22 se presenta un diagrama de distribución que representa la arquitectura del sistema posterior a la integración de los plugins `XmlGen` y `XmlImport`.

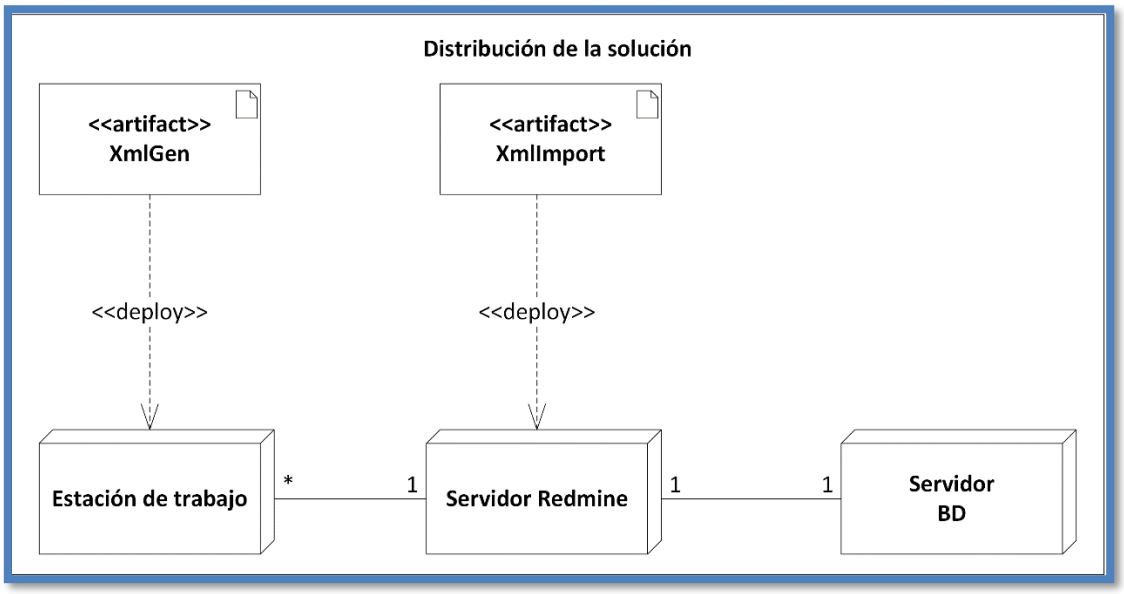


Figura 22: Diagrama de la distribución de la solución

Dada la naturaleza de la solución, que consiste en dos plugins que funcionan en una herramienta (EPFC) y en una plataforma web (Redmine), se recomienda tener al menos dos entornos de sistema distintos ya sean físicos o virtuales. Cabe destacar que la solución es independiente de los entornos instalados (p.ej. Windows, OSX, Linux) ya que sólo dependen respectivamente de EPFC y Redmine.

Por otro lado, para el plugin `XmlImport`, se requiere solamente de la plataforma Redmine instalada y de ningún otro plugin externo. Redmine, al ser una plataforma popular dentro del mundo del desarrollo de software, tiene y existen múltiples formatos en los cuales se puede instalar; por lo tanto, ésta posee mayor flexibilidad en cuanto al entorno en el que se puede instalar.

En la Tabla 6 se describen todos los componentes y requisitos necesarios para poder ejecutar el sistema con un correcto funcionamiento.

Componente	Requiere
Estación de trabajo	Sistema operativo compatible con EPFC Eclipse Process Framework Composer v1.5.1.7 Web Browser (p.ej. Chrome, IE, Firefox) XmlGen
Servidor Redmine	Sistema operativo compatible con Redmine Redmine 3.1.1 Ruby 2.0 Ruby on Rail 4.2.2 MySQL 5.5 XmlImport

Tabla 6: Requisitos del sistema

5 Implementación de la solución

En esta sección se describe la visualización y uso de los distintos plugins creados para el sistema. A modo de ejemplificar sus usos se utilizó un proceso formalizado por Ki Teknology.

5.1 Extracción de la información del proceso (CU1)

`XmlGen` interactúa con el programa EPFC para la exportación del proceso modelado mediante un archivo XML. Para facilitar el uso de `XmlGen`, se generó un wizard el cual, paso a paso, va guiando al usuario para la exportación del proceso y la correcta generación del artefacto de intercambio XML.

Al inicializar EPFC, el plugin agrega una opción en la barra superior del programa, como se puede ver en la Figura 23.

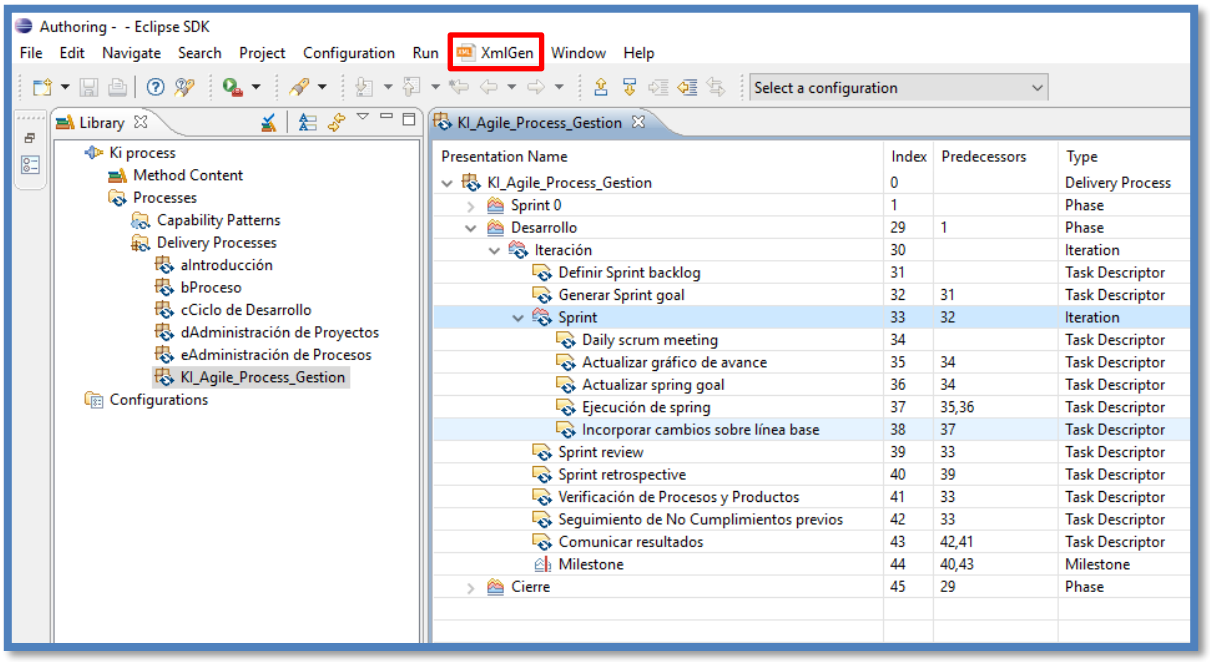


Figura 23: Vista de EPFC en donde se encuentra XmlGen

Hay que enfatizar que para poder hacer uso correcto de `XmlGen` hay que tener el proceso modelado en EPFC de la manera descrita en la Sección 2.1.4. Se observa en la Figura 23 dentro de la pestaña 'Library' que el Method Plugin utilizado es de Ki Teknology y se llama Ki process.

El siguiente paso es seleccionar la opción `XmlGen` en la barra, la cual despliega el wizard y muestra los Method Plugins disponibles para elegir. Una vez seleccionado, despliega los Delivery Process que el Method Plugin seleccionado contiene en donde el usuario debe seleccionar qué proceso quiere que se genere en el archivo XML. Se muestra en la Figura 24, el proceso de selección.

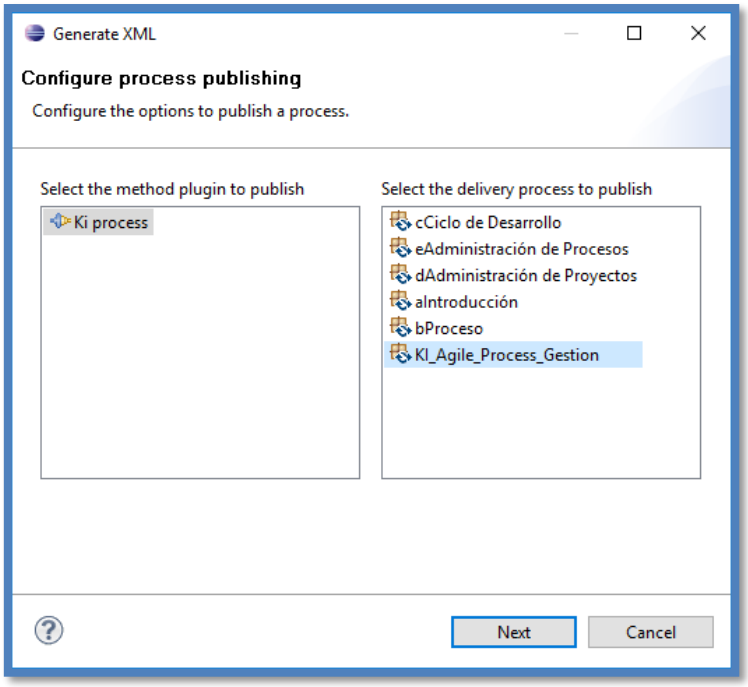


Figura 24: Ventana de selección de proceso

Para continuar, el usuario debe seleccionar 'next'. Aquí el plugin analiza si el proceso contiene o no iteraciones para así, en la siguiente visualización, desplegar primero el campo 'Project name', que el usuario debe completar. Finalmente se despliegan los campos para cada iteración (en caso de que hubiera), que el usuario debe completar. En este ejemplo se encontraron dos: 'Desarrollo : Iteración' e 'Iteración : Sprint'. La Figura 25 muestra la interfaz que el usuario visualiza y en dónde debe completar los campos requeridos.

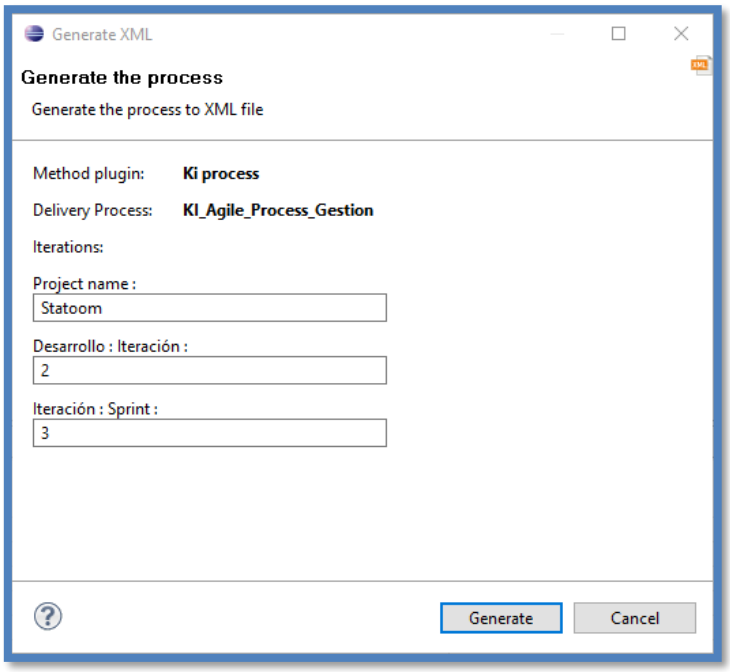


Figura 25: Segunda página de la interfaz de XmlGen

El usuario, habiendo completado todos los campos, selecciona 'generate' y el plugin creará un archivo XML, siguiendo el esquema expuesto en la Sección 4.2.3, con toda la información del proceso. Finalizado el proceso se despliega una interfaz para que el usuario guarde el archivo en el entorno en que se encuentra. La Figura 26 muestra la interfaz de guardado del archivo generado.

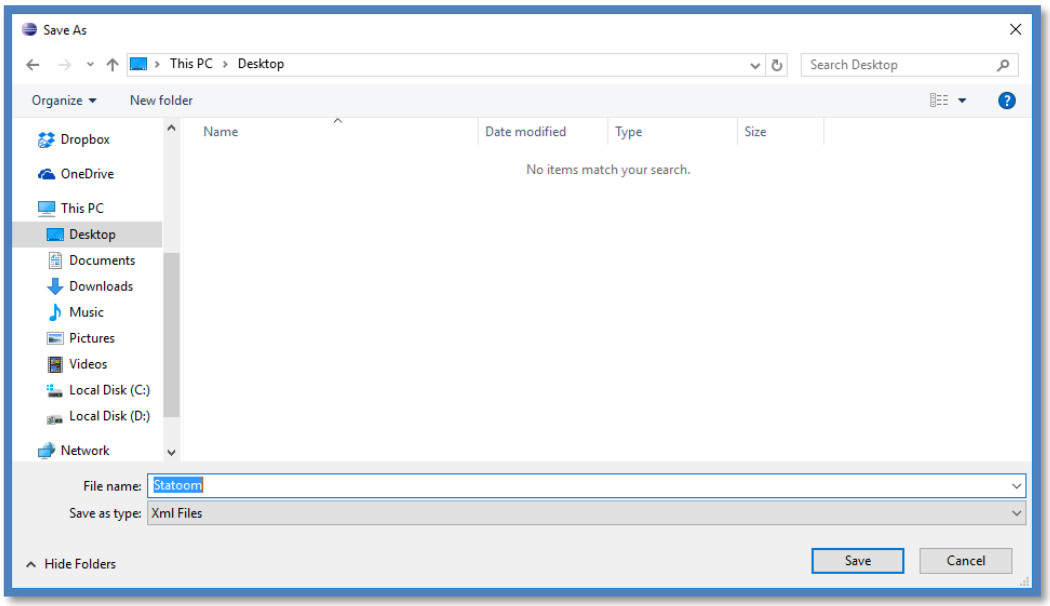


Figura 26: Browser del directorio para guardar el archivo XML

Finalmente, una vez guardado el archivo, se despliega la última página del wizard, en la que muestra un mensaje de confirmación de que la exportación fue un éxito. En la Figura 27 se puede apreciar la última interfaz de éxito de `XmlGen`.

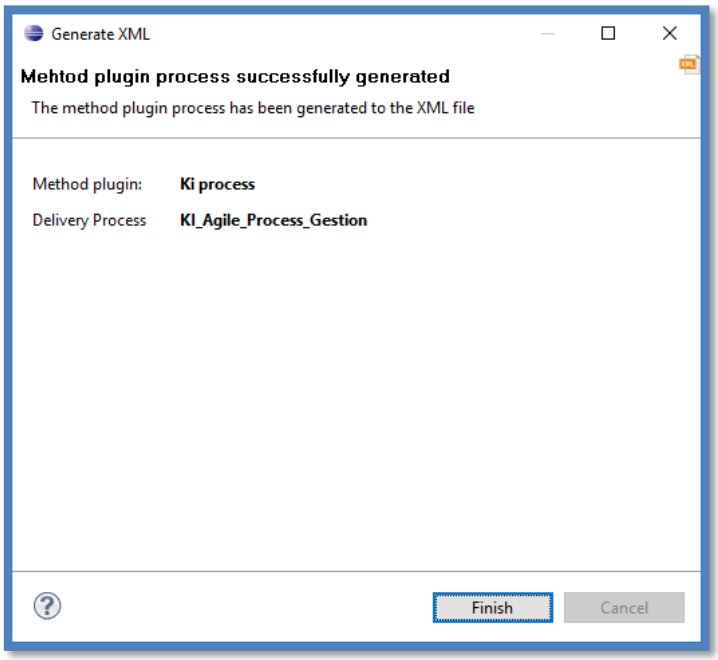


Figura 27: Tercera página de la interfaz de XmlGen

En este punto, el proceso de creación del archivo XML con la información del proceso ha finalizado. Ahora el usuario deberá ingresar, mediante un explorador web (p.ej. Chrome, IE, Firefox), a la plataforma Redmine para hacer uso del archivo XML generado por `XmlGen`.

5.2 Generar plan de desarrollo a partir de la información del proceso (CU2)

`XmlImport` es el plugin que se instala en la plataforma Redmine. Su propósito es importar la información y crear un plan de desarrollo a partir de un archivo XML que contenga la información del proceso. En la Figura 28 se observa dónde se ubica el vínculo que el plugin aporta a la plataforma Redmine.

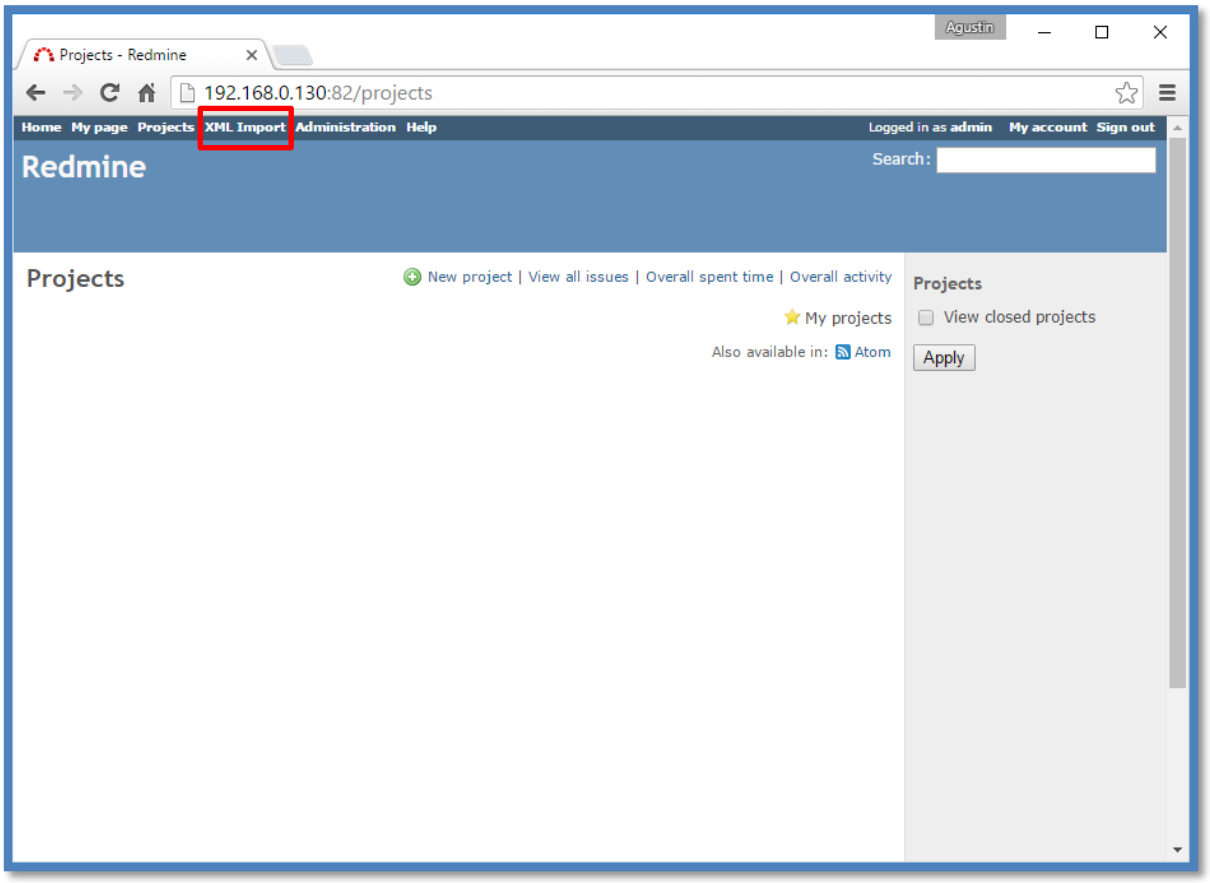


Figura 28: Vista de la plataforma Redmine con XmlImport

El usuario selecciona la opción 'XmlImport' y se despliega la vista para escoger el archivo a importar. Aquí, el usuario debe seleccionar el archivo XML generado por XmlGen, como se ilustró en la sección 4.2.3, para luego seleccionar 'import'. La Figura 29 muestra la interfaz de selección e importación del archivo XML.

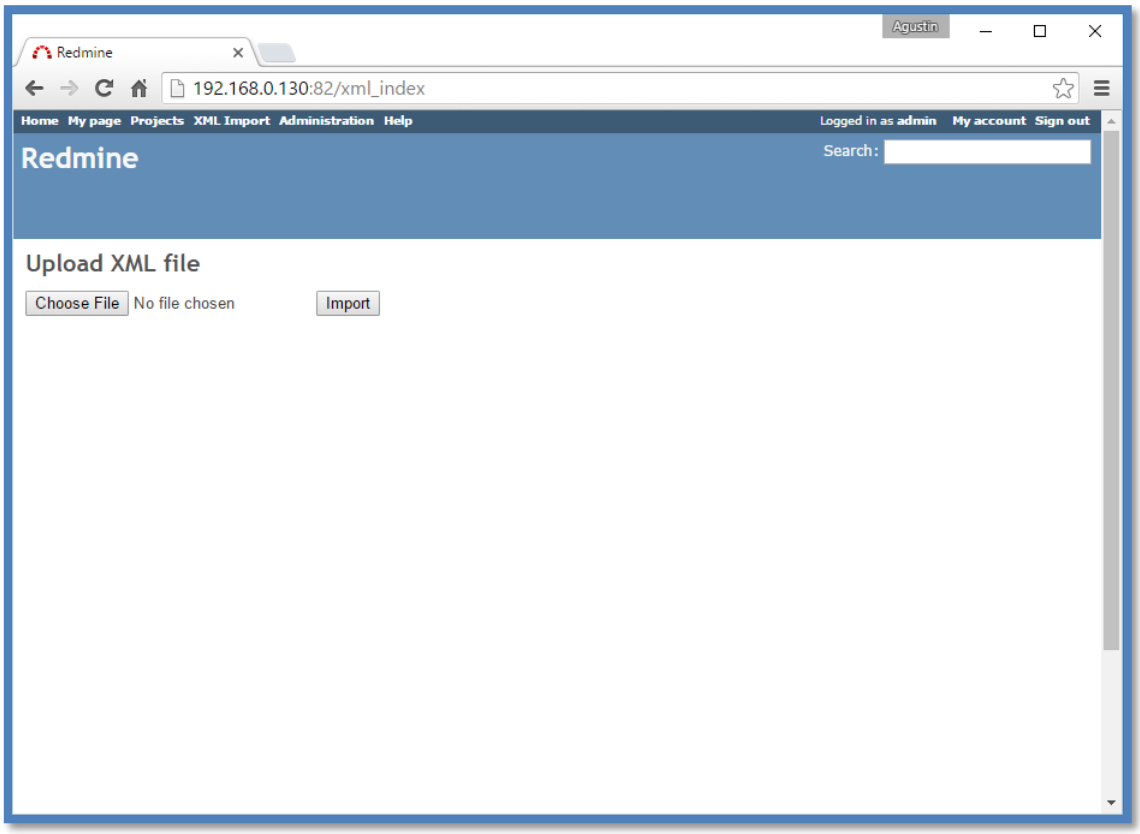


Figura 29: Interfaz de selección e importación

Una vez seleccionado el archivo y que el usuario haya seleccionado 'import', el plugin analiza el archivo XML y comienza el proceso de generación del plan de desarrollo. Una vez finalizado dicho proceso, el plugin redirige al usuario a la lista de proyectos que están en la plataforma, desplegando un mensaje de éxito si fuese el caso, de lo contrario arroja un error. En la Figura 30, se muestra el proceso de importación finalizado. Además, la figura muestra también el nuevo proyecto 'Statoom' en el listado de proyectos, el cual no estaba presente en la Figura 28 y fue creado al utilizar el plugin `XmlImport`.

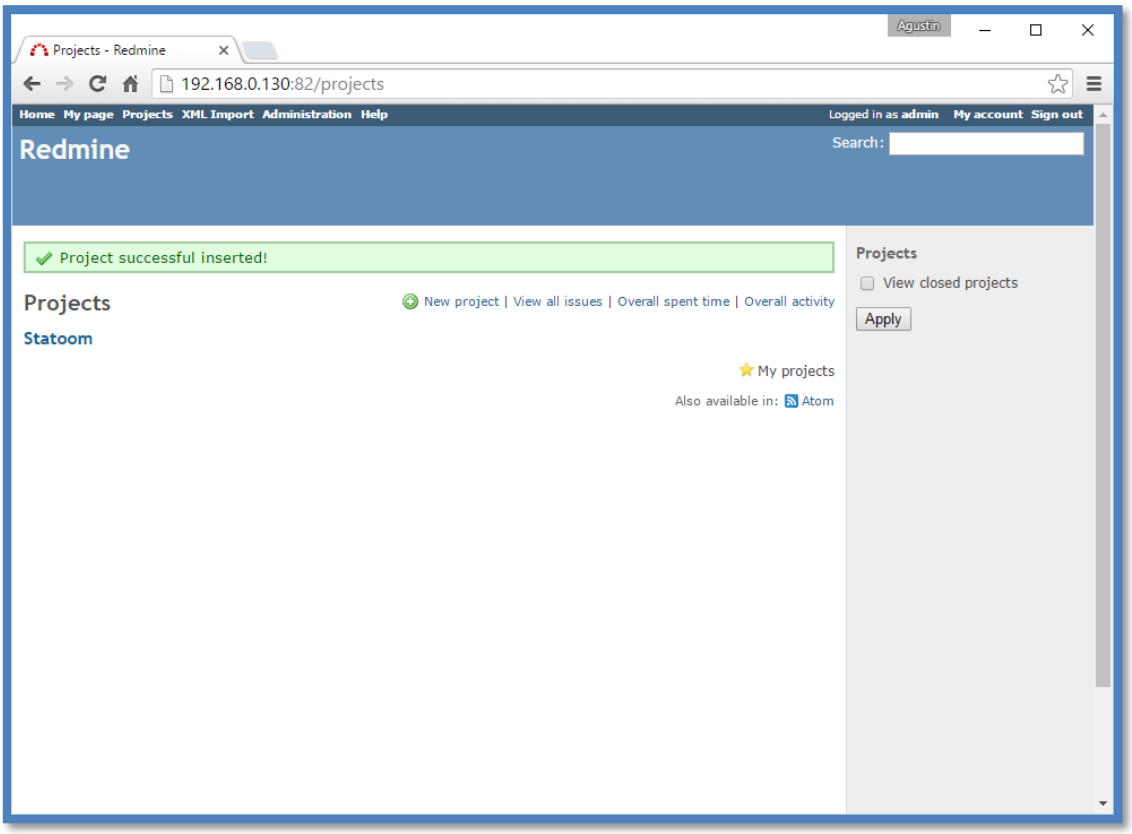


Figura 30: Interfaz de éxito de XmlImport

Finalmente, en la Figura 31, podemos ver el resultado de todo el proceso del sistema. Es decir, a partir de un proceso en EPFC exportado a un artefacto de intercambio y luego importado mediante un plugin a la plataforma Redmine, se puede ver la carta Gantt de un plan de desarrollo con sus distintas tareas.

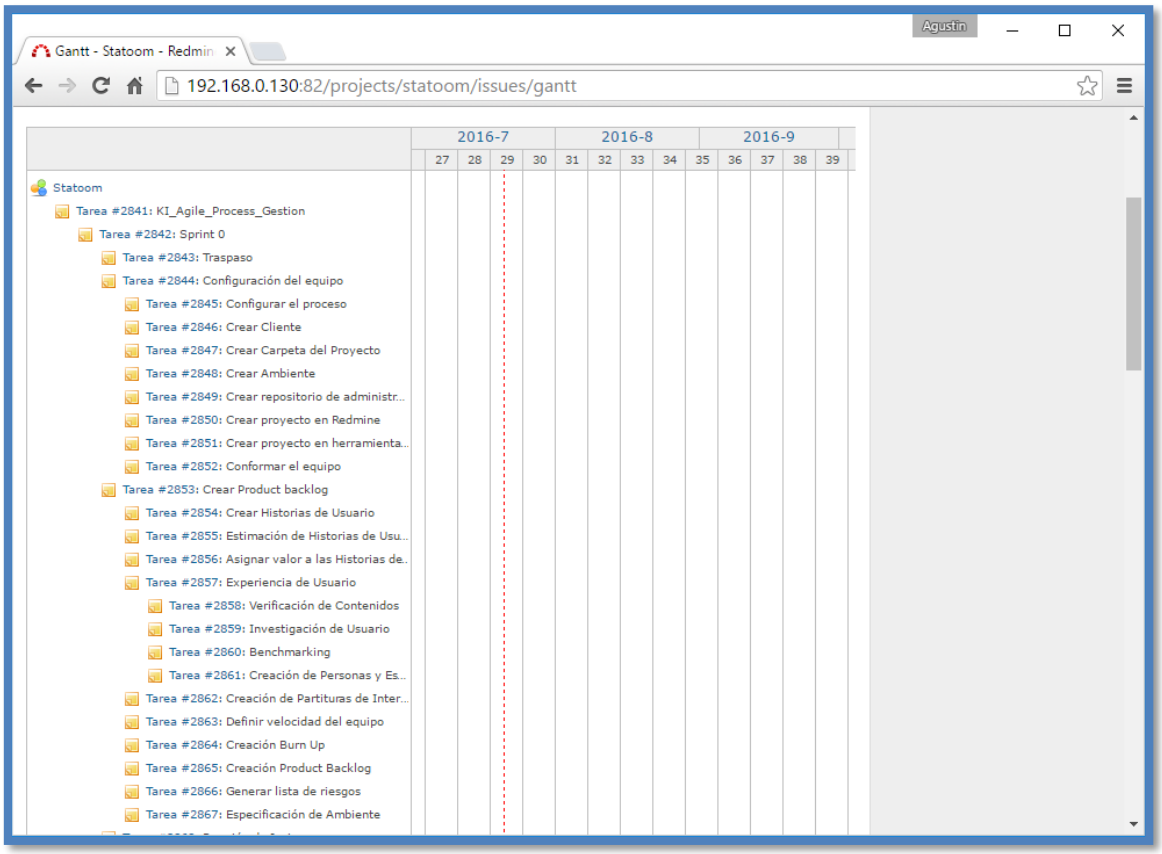


Figura 31: Carta Gantt resultante de la inserción de un proceso

Se observa en la Figura 31, que la carta Gantt resultante de la inserción de un proceso mediante `XmlImport` no muestra las precedencias entre las tareas del proceso, a pesar que éstas contengan dicha información. Lo anterior se debe a que ninguna tarea posee una fecha de inicio ni de término. Si editamos una tarea cualquiera y le damos fecha de inicio y término, Redmine recalcula la duración de todas las tareas que dependan de ésta y genera una nueva carta Gantt mostrando las precedencias de las tareas que fueron creadas por el plugin `XmlImport`. En la Figura 32 se aprecia el resultado cuando se modifica la duración de una tarea.

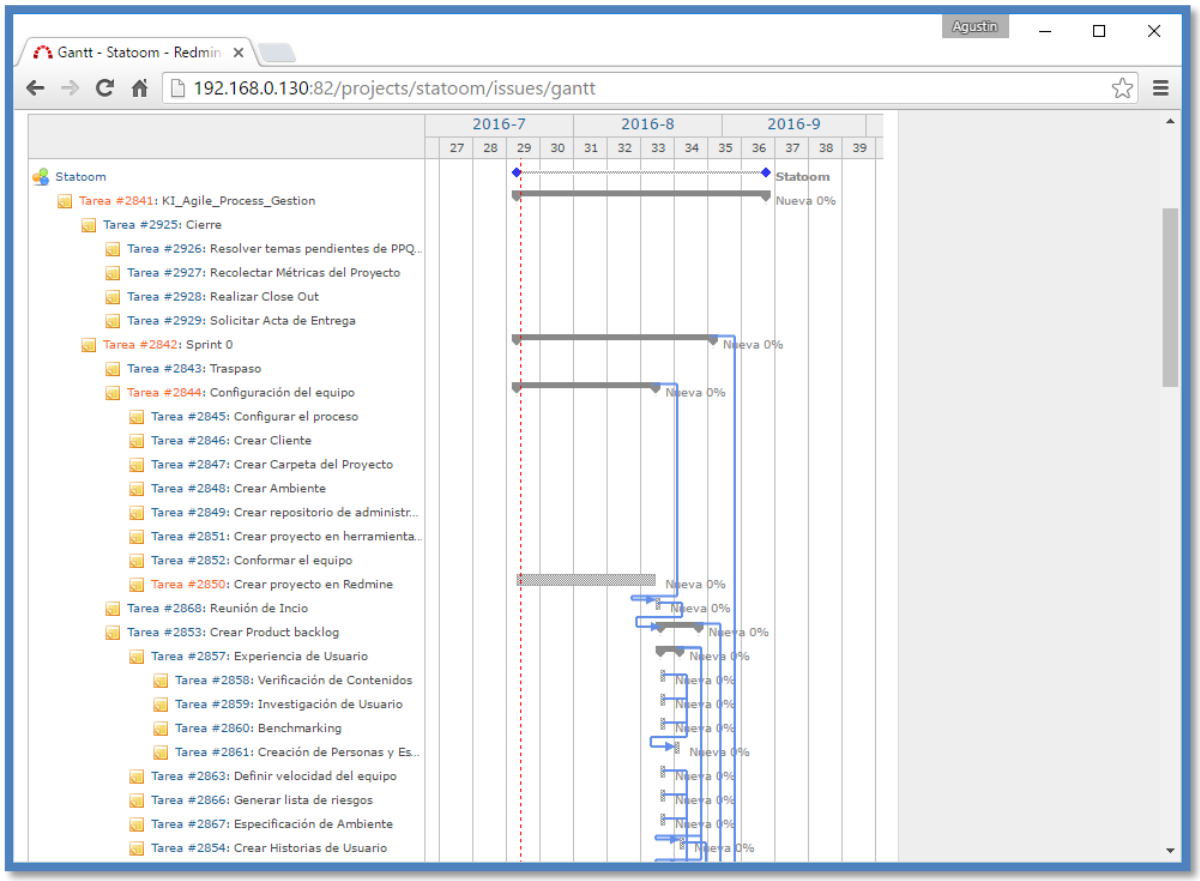


Figura 32: Carta Gantt resultante al agregar duración de una tarea

Finalmente, en la Figura 33 podemos ver el resultado dentro de un plan de desarrollo, generado mediante `XmlGen`, de cómo se duplican las iteraciones de un proceso.

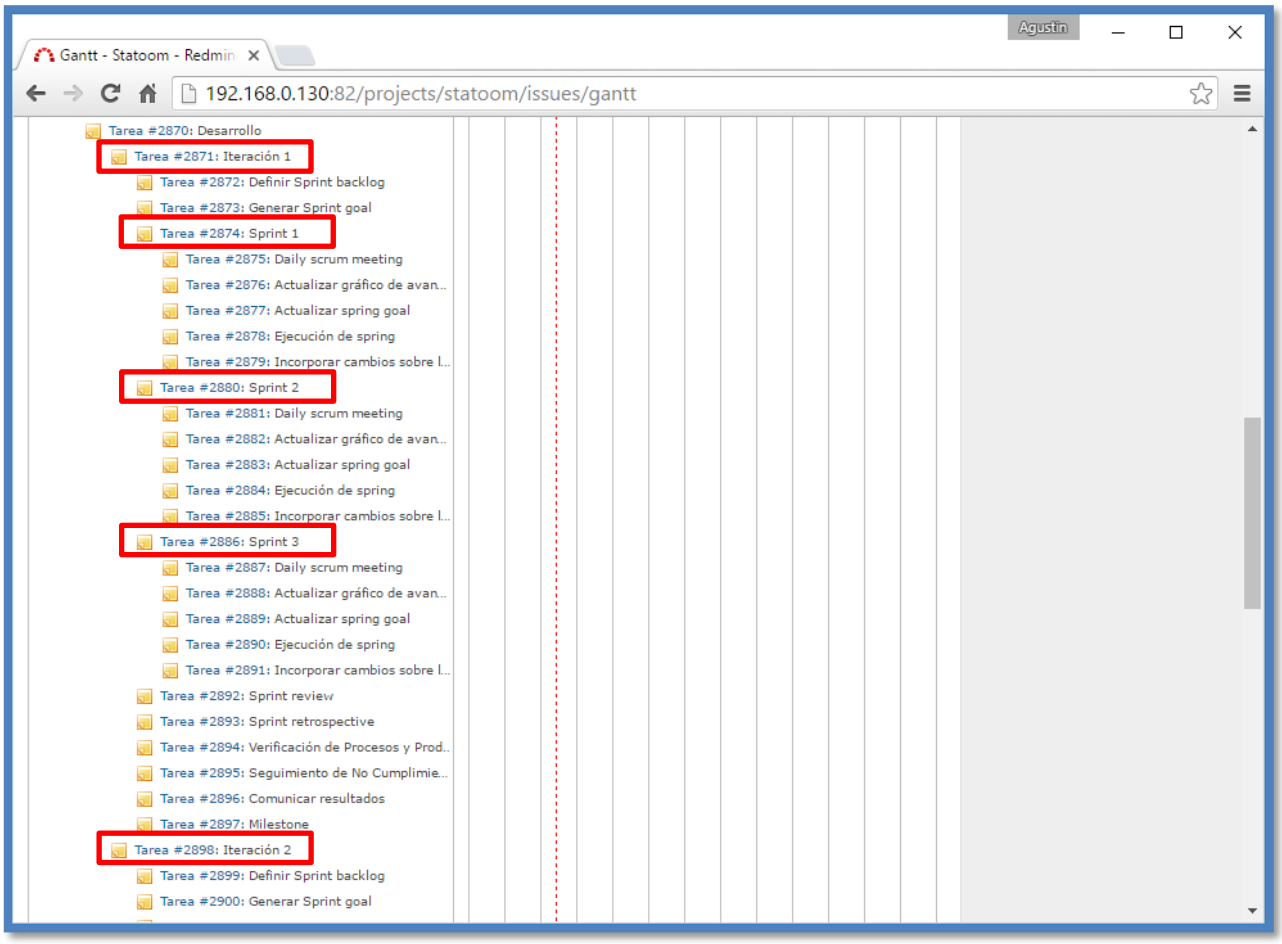


Figura 33: Vista de las iteraciones en un plan de desarrollo

6 Validación de la solución

En este capítulo se detalla el proceso que se llevó a cabo para validar la solución propuesta en este trabajo de memoria. Para ello, se expuso la solución a dos paneles, uno formado por usuarios expertos de la empresa Ki Teknology, éste constituye el cliente directo de esta solución; y otro de expertos del proyecto GEMS perteneciente al DCC de la Universidad de Chile.

6.1 Validación con usuarios expertos de Ki Teknology

Con el fin de validar la solución propuesta en este trabajo, se realizó una reunión con la empresa Ki Teknology en sus dependencias. A esta reunión asistieron Matías Flores, usuario experto en el uso de planes de trabajo dentro de la plataforma Redmine, Daniel Perovich en su rol de investigador asociado del proyecto GEMS y Agustín López en su rol de desarrollador validador de la solución. La reunión se dividió en dos partes, en la primera se expuso acerca del contexto del problema y además se presentó la solución y las limitaciones de ella.

En la segunda parte de la reunión, se hizo una demostración de ambos componentes del sistema. En primer lugar, se demostró el funcionamiento de `XmlGen` dentro de la herramienta EPFC. Para ello se usó un proceso modelado por Ki Teknology, tarea en la cual había participado Matías Flores, pero adaptado según las consideraciones de la Sección 3.2.2. A partir de la demostración se obtuvo un archivo XML que contenía toda la información del proceso.

En segundo lugar, se demostró el funcionamiento del plugin `XmlImport` dentro de la plataforma Redmine. Para ello se utilizó el archivo XML obtenido anteriormente en la demostración del plugin `XmlGen`.

Finalizadas las demostraciones de ambos plugins, se le pidió al usuario de Ki Teknology que informara sus consideraciones respecto a la solución, las que son detalladas a continuación:

VALIDACIÓN

- Ambos plugins son de fácil uso.
- El sistema es de gran utilidad y para ellos es necesario una solución con estas características.
- Ésta les ahorraría mucho tiempo y esfuerzo y además les reduciría los errores dado que hoy el proceso de generación de un plan de desarrollo es manual.
- Para poner el sistema en producción es necesario que la solución se haga cargo de la adaptación del proceso formalizado.
- Encontraron de gran utilidad el artefacto de intercambio ya que, si lo necesitasen, podrían adaptarlo a sus necesidades para luego importarlo a Redmine.

NUEVAS CARACTERÍSTICAS DESEABLES

- Añadir la capacidad de generar fragmentos o partes del proceso en el plugin `XmlGen` para luego poder importarlos a Redmine mediante `XmlImport`.
- Añadir la característica de asignación de duración a las tareas, actividades e iteraciones de tiempo total y duración promedio en `XmlGen`.
- Integrar este sistema con la herramienta de adaptación de procesos desarrollada por el equipo GEMS.

6.2 Validación con el panel de expertos

Con el fin de validar la solución propuesta en este trabajo, se realizó una reunión con los integrantes del proyecto GEMS. Ésta fue en las dependencias del Departamento de Ciencias de la Computación situada en la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. A esta reunión asistieron siete miembros del equipo del proyecto GEMS, entre ellos directores, investigadores y tesistas.

El formato de la reunión fue exactamente igual al usado en la reunión para presentar la solución a la empresa Ki Teknology. Finalizada la reunión se les pidió a los asistentes que expusieran sus consideraciones. A continuación, se detallan:

VALIDACIÓN

- Resulta de gran utilidad que exista un artefacto intermedio que contenga toda la información del proceso, esto permite a futuro que otros programas puedan consumirlo.
- Se pierde información sobre las condiciones cuando existen ciclos dentro de una actividad en un proceso.
- Se encontró útil, de fácil uso y que cumple con los objetivos de esta herramienta.

- Se propuso que se publicaran las soluciones de manera open source a la comunidad de EPFC y Redmine.

NUEVAS CARACTERÍSTICAS DESEABLES

- Sería de gran utilidad agregar duración de iteraciones para el plugin `XmlGen`.
- Agregar soporte de milestones al plugin `XmlImport`. Es decir, que sea capaz de diferenciar entre una tarea y un milestone.

6.3 Análisis de los resultados

A partir de las consideraciones de ambos paneles de expertos, se concluye que la solución construida logra satisfacer la problemática de generar automáticamente planes de desarrollo a partir de procesos formalizados. A su vez, ambos paneles se mostraron motivados a agregar funcionalidades adicionales al sistema y además identificaron las limitaciones de éste.

Un elemento destacable del sistema fue la implementación de un artefacto intermedio de intercambio de información. Ambos paneles estuvieron de acuerdo que, gracias a la arquitectura implementada, el sistema pose la característica de ser extensible a otras plataformas de gestión de proyectos de desarrollo.

Sin embargo, hay que destacar que si el proceso pose ciclos, al momento de generar el plan de desarrollo mediante `XmlImport`, el plan perderá la información asociada a la condicionalidad que generaba ese ciclo. Hay que recordar, que la posible pérdida de información al generar un plan de desarrollo desde un proceso modelado en EPFC, surge del nivel de expresividad que existe entre el WBS y el diagrama de actividad de las tareas.

Otra limitación fue la necesidad de partir de un proceso adaptado del original. Sin embargo, esta limitación no está dentro del alcance de este trabajo de memoria. Esta limitación la abarca una herramienta perteneciente al proyecto GEMS como se muestra en la Figura 1. De todas maneras, a pesar de esta limitación, la solución es capaz de reducir el costo de tiempo y esfuerzo en la generación de un plan de desarrollo para un proyecto. Finalmente, si se necesitase aún así de un proceso adaptado, el ingeniero de procesos puede adaptar el proceso manualmente y luego utilizar el sistema para la generación del plan de desarrollo.

Una tercera limitación es no poder asignar duración a las actividades. Una posible mejora seria poder asignar duración a los milestones al momento de generar el artefacto de intercambio, para que así, al crear el plan de desarrollo en Redmine, este tuviese duración. Sin embargo, Redmine trata un milestone como una tarea común y corriente por lo tanto no le da el carácter de hito que el usuario podría esperar.

Otra estrategia para la asignación de duración y miembros del equipo de desarrollo a actividades, es utilizar la información histórica del proyecto. Así el sistema podría asignar dinámicamente la duración y quien debe realizar la actividad a partir de la información histórica registrada. Sin embargo, dicha funcionalidad está fuera del alcance de esta memoria, pero representa una línea de trabajo relevante para la evolución de este sistema.

Hay que tomar en consideración que esta es una primera aproximación al problema propuesto. Es por ello, que esta solución no considera ciertos aspectos relevantes, mencionados con anterioridad, por parte de los paneles de validadores del sistema.

7 Conclusiones y trabajos futuros

A lo largo de este trabajo se estudió un amplio espectro de información y herramientas, partiendo desde el análisis de la situación actual de la creación de planes de desarrollo a partir de procesos formalizados, hasta la implementación de herramientas tecnológicas para la automatización de ellos.

Teniendo en cuenta los diferentes factores asociados al problema que enfrentan las PyMEs de desarrollo en Chile, se estudiaron diversos enfoques para darle una solución tecnológica a la creación y generación de planes de desarrollo a partir de sus procesos formalizados. A partir de lo anterior, se optó por enfocarse en dos tecnologías open source. Una de ellas, Eclipse Process Framework Composer (EPFC), herramienta utilizada para formalizar procesos. La otra, Redmine, plataforma dedicada a la gestión de proyectos de desarrollo y ampliamente utilizada en la industria de desarrollo de software.

Se ideó una solución, que consiste en integrar al funcionamiento de las herramientas elegidas, plugins desarrollados con ciertas funcionalidades. Uno de ellos se enfoca en analizar y extraer la información del proceso desde EPFC a un artefacto de intercambio y el otro en importarlo exitosamente hacia la plataforma Redmine. La solución construida resultó de gran interés y utilidad, tanto en el contexto del proyecto de investigación GEMS, como en los usuarios finales de las PyMEs con las que se trabajó.

Además cabe destacar que el sistema desarrollado puede utilizarse tal como está, logrando así reducir errores y el esfuerzo que nace de la generación manual de un plan de desarrollo. Finalmente, a partir de la validación, surgieron nuevas necesidades de integración con herramientas del proyecto GEMS. Lo que demuestra que este sistema puede insertarse al paquete actual de herramientas del proyecto de investigación del DCC.

Dentro de los mayores desafíos enfrentados durante el desarrollo de este trabajo estuvo la integración con herramientas existentes. Es decir, hubo un extenso estudio previo a la implementación de la solución con el fin de lograr el correcto actuar de las partes de la solución con sus respectivas herramientas. El desafío radicaba en que las herramientas no disponen de documentación detallada de ciertos aspectos esenciales para el desarrollo de la solución.

No obstante, el gran reto que estas situaciones representaron en la realización del proyecto de memoria, fueron paralelamente una gran fuente de aprendizaje en relación a cómo este tipo de factores impactan en el desarrollo de proyectos dentro del rubro tecnológico.

Considerando todo lo anteriormente mencionado, se puede concluir que el objetivo general del trabajo de título fue alcanzado. Se logró mejorar el proceso de creación de planes de desarrollo a partir de procesos formalizados mediante herramientas open source. En un futuro esto le permitirá a las PyMEs de desarrollo agilizar y obtener planes de desarrollo más fidedignos al proceso modelado para el desarrollo de un software.

8 Bibliografía

[1] J. Lonchamp, «A structured conceptual and terminological framework for software process engineering,» Second international conference on the continuos software process improvement, 1993, pp. 41-53.

[2] W. S. Humphrey, Managing the Software Process, Addison-Wesley, 1989.

[3] J. A. Hurtado Alegría, M. C. Bastarrica y A. Bergel, «AVISPA: a tool for analyzing software process models,» *Journal of software: Evolution and process,* nº 26, pp. 434-450, 2014.

[4] L. Silvestre, C. Bastarrica y S. Ochoa, «A usable MDE-based tool for software process tailoring,» de *ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems*, Ottawa, Canada, Septemter 27 - October 2 (2015).

[5] DCC, «GEMS,» Departamento de Ciencias de la Computacion - Universidad de Chile, 2016. [En línea]. Available: http://dcc.uchile.cl/gems/. [Último acceso: Mayo 2016].

[6] C. Bastarrica, «Análisis estadístico de la encuesta de caracterización de las PyMEs de software chilenas,» 7 Octubre 2015. [En línea]. Available: http://www.dcc.uchile.cl/gems/docs/InformeAnalisisEncuesta2015.pdf.

[7] E. Foundation, «Eclipse Process Framework Project,» Eclipse Foundation, [En línea]. Available: http://projects.eclipse.org/projects/technology.epf.

[8] Object Managment Group, «Software & Systems Process Engineering V2.0,» April 2008. [En línea]. Available: http://doc.omg.org/formal/08-04-01.pdf. [Último acceso: June 2016].

[9] I. Sommerville, Software Engineering 9th, Pearson, 2011.

[10] Project Managment Institute, A Guide to the Project Management Body of Knowledge 4th edition, 2008.

[11] B. Tuft, «Eclipse Process Framework (EPF) Composer - Installation, Introduction, Tutorial and Manual,» February 2010. [En línea]. Available: http://epf.eclipse.org/uploads/14.pdf.

[12] J.-P. Lang, «Redmine wiki,» Redmine, 2014. [En línea]. Available: https://www.redmine.org/projects/redmine/wiki.

[13] Solid IT, «Project Managmente Zone,» 25 June 2016. [En línea]. Available: http://project-management.zone/ranking/open-source,planning. [Último acceso: 15 March 2016].

[14] easy REDMINE, «Easy Redmine Clients and Case Studies,» easy REDMINE, [En línea]. Available: https://www.easyredmine.com/clients?demoversion=b.

[15] J. Niggemann, «Who uses Redmine?,» Redmine, 03 04 2016. [En línea]. Available: http://www.redmine.org/projects/redmine/wiki/WeAreUsingRedmine.

[16] Object Managment Group, «Unified Modeling Language: Infrastructure,» 07 05 2005. [En línea]. Available: http://doc.omg.org/formal/2005-07-05.pdf.

[17] N. Rozanski y E. Woods, Software Systems Architecture - Working with Stakeholders Using Viewpoints and Perspectives, Addison - Wesley, 2009.