



UNIVERSIDAD NACIONAL DE COLOMBIA

OBTENCIÓN DE DIAGRAMAS BPMN CON RECURSOS HUMANOS A PARTIR DE PROCESOS DE SOFTWARE EN SPEM 2.0

DIEGO ESTEBAN CRUZ ROJAS

UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS E INDUSTRIAL
BOGOTÁ - COLOMBIA
2013

OBTENCIÓN DE DIAGRAMAS BPMN CON RECURSOS HUMANOS A PARTIR DE PROCESOS DE SOFTWARE EN SPEM 2.0

DIEGO ESTEBAN CRUZ ROJAS

Tesis de investigación presentada como requisito parcial para
optar al título de:

MSc. en Ingeniería de Sistemas y Computación

Directora:

HELGA DUARTE-AMAYA, Ph.D

Profesora Asociada

Departamento Ingeniería Sistemas e Industrial

Universidad Nacional De Colombia

Asesora:

MARÍA CECILIA BASTARRICA, Ph.D

Profesora Asociada

Departamento De Ciencias De La Computación

Universidad De Chile

Línea de Investigación:

Ingeniería de Software

UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS E INDUSTRIAL
BOGOTÁ - COLOMBIA
2013

A Dios y a mis ángeles que desde el
cielo también configuran mis Bendiciones en
la tierra.

Agradecimientos

Agradezco a la Facultad de Ingeniería de la Universidad Nacional de Colombia y a sus docentes por haber contribuido ampliamente a mi formación académica. En particular al profesor Jorge Eduardo Ortiz, quien motivó la participación en el programa.

A la profesora Helga Duarte, directora de esta tesis por su orientación y sus valiosas sugerencias que hicieron posible el buen término de este proyecto.

A la profesora Cecilia Bastarrica, del departamento de Ciencias de la Computación de la Universidad de Chile, por su infinita colaboración.

Al Fondo de Fomento al Desarrollo Científico y Tecnológico de Chile (FONDEF) que a través del proyecto ADAPTE financió parcialmente el presente trabajo.

A mí siempre amigo Alejandro Vivas, quien supo encontrar salidas técnicas cuando todo parecía perdido.

Resumen

Las empresas de software definen sus procesos de desarrollo como una forma de organizar y planear sus actividades, y como un medio para alcanzar una posible certificación ISO o una evaluación CMMI. Existen además estándares como SPEM y herramientas libres como EPF Composer, que permiten que toda empresa pueda contar con un soporte robusto y accesible para la definición de sus procesos de desarrollo de Software. La definición de un proceso de software y su implantación en la organización es una actividad que demanda bastantes recursos en términos de conocimiento, tiempo y costo, y muchas veces los procesos allí definidos no se aplican en la práctica porque resulta complejo para el equipo de desarrollo. De otro lado, BPMN se ha convertido en un estándar para la definición de procesos de negocio, que proporciona claridad gráfica y la posibilidad de que los procesos especificados bajo éste puedan ser gestionados de manera automática sobre una plataforma BPMS. Transformar los procesos de desarrollo de software en SPEM a procesos de negocio en BPMN implicaría un menor nivel de especificidad, pero permitiría acercarse a la gestión automatizada de procesos de software. Adicionalmente, BPMN no provee soporte para la representación de estructuras de recursos humanos, lo cual impone una dificultad para asignar directamente las tareas al grupo de trabajo. En el presente trabajo proponemos un enfoque basado en MDE (*Model Driven Engineering*) para la transformación automática de procesos en SPEM a procesos BPMN relacionando los roles responsables de la ejecución de cada tarea. La transformación implementada en XSLT se valida sobre un proceso de desarrollo de software real, una empresa chilena llamada Mobius.

Palabras clave: proceso de software, proceso de negocio, asignación de recursos humanos, desarrollo dirigido por modelos

Abstract

Software companies define their development processes as a means for organizing their activities, and also to make it possible to achieve an ISO certification or a CMMI evaluation. There are standards such as SPEM y free tools as EPF Composer, that allow companies to count on robust and available support for defining their software processes. However, this definition is a laborious and expensive task, and even so the defined processes are sometimes not applied in practice because developers find them complex and cumbersome. On the other hand, BPMN has become a de facto standard for defining business processes, providing a clear graphical representation and the possibility to automatically transform these processes in order to execute them on a BPMS platform.

Nevertheless, BPMN is not specifically defined for software processes y thus is not as expressive as SPEM for these purposes. Additionally, BPMN does not support the representation of human resources so it is difficult to directly assign tasks to a work team. In this work we propose an MDE-based approach for automatically transforming SPEM processes into BPMN processes relating tasks with the roles responsible for their execution. The transformation has been implemented using XSLT and validated on the software development process of a real Chilean company called Mobius.

Keywords: software process, business process, human resource assignment, Model-Driven Development

Contenido

	<u>Pág.</u>
Resumen	VI
Abstract	VII
Lista de figuras	X
Lista de tablas	XII
Lista de Símbolos y abreviaturas	XIII
Capítulo 1. Introducción	15
1.1 Proceso de Desarrollo de Software	15
1.2 Gestión de Procesos de Negocio BPM.....	16
1.3 Arquitectura Dirigida por Modelos	17
1.4 Problema a Resolver.....	18
1.5 Solución Propuesta.....	20
1.6 Objetivos de la tesis.....	21
1.6.1 Objetivo General.....	21
1.6.2 Objetivos Específicos.....	22
1.6.3 Resultados Esperados.....	22
1.7 Estructura de la Tesis.....	22
Capítulo 2. Background & Trabajos Relacionados	23
2.1 Unified Method Architecture (UMA)	23
2.1.1 Principales aportes de UMA a SPEM.....	24
2.2 Software Process Engineering Metamodel (SPEM)	27
2.2.1 Características principales de SPEM.....	27
2.3 Eclipse Process Framework Composer (EPF Composer)	31
2.3.1 Arquitectura de EPF Composer.....	32
2.3.2 Estándares usados en EPF Composer.....	34
2.3.3 Relación de versiones de EPF Composer & UMA.....	34
2.4 Business Process Model and Notation (BPMN)	35
2.4.1 Diagramas BPMN.....	35
2.4.2 Limitaciones de BPMN.....	38
2.5 Transformación de Modelos.....	39
2.5.1 Lenguaje XSLT para transformación de modelos.....	40
2.6 Trabajos relacionados.....	41
2.6.1 Transformación de Modelos de Procesos de Desarrollo de Software a Modelos de Procesos de Negocio	41

2.6.2	Metamodelos de Recursos Humanos	45
Capítulo 3.	Desarrollo de la Solución.....	49
3.1	Metamodelo de Recursos Humanos	50
3.2	Modelos Fuente -Procesos de Desarrollo en <i>SPEM</i>	51
3.3	Diseño de la Transformación de Modelos <i>SPEM</i> a <i>BPMN</i>	53
3.4	Implementación de la Transformación de Modelos <i>SPEM</i> a <i>BPMN</i> 55	
3.5	Modelos Objetivo - Procesos de Negocio en <i>BPMN</i>	57
Capítulo 4.	Validación de la Transformación	60
4.1	Caso de Estudio.....	61
4.2	Restricciones del experimento.....	61
4.3	Transformación del Proceso de Desarrollo de Software de la empresa Mobius.....	62
4.3.1	Transformación de la actividad <i>Análisis de Requerimientos</i>	62
4.3.2	Transformación de la actividad <i>Planificación del Proyecto</i>	64
4.3.3	Transformación de la actividad <i>Acuerdo Comercial</i> ...	66
4.3.4	Transformación de la actividad <i>Lanzamiento</i>	69
4.3.5	Transformación de la actividad <i>Validación de Arquitectura</i>	69
4.3.6	Transformación de la actividad <i>Control de Cambios</i> ..	71
4.3.7	Transformación de la actividad <i>Diseño de Pruebas</i> ...	74
4.3.8	Transformación de la actividad <i>Desarrollo de Solución</i> 75	
4.3.9	Transformación de la actividad <i>Ejecución de Pruebas</i>	78
4.3.10	Transformación de la actividad <i>Liberación</i>	79
4.3.11	Transformación de la actividad <i>Implementación</i>	80
Capítulo 5.	Conclusiones y Recomendaciones	83
5.1	Resumen del trabajo realizado.....	83
5.2	Revisión de Objetivos.....	84
5.3	Principales Contribuciones.....	85
5.4	Lecciones Aprendidas.....	86
5.5	Trabajo Futuro.....	87
		89
A.	Código fuente de la transformación.....	91
B.	Bibliografía	99

Lista de figuras

	<u>Pág.</u>
Figura 1 Esquema de la solución planteada	21
Figura 2 Principales modelos que dan origen a UMA [35]	24
Figura 3 Elementos principales de UMA [36]	25
Figura 4 Paquetes del metamodelo de SPEM 2.0. [2]	28
Figura 5 Niveles de modelado e Instanciación MDA	29
Figura 6 Arquitectura EPF Composer [33]	32
Figura 7 Trabajos relevantes en la Transformación de Modelos ...	42
Figura 8 Transformación de SPEM a BPMN con MOSKitt4ME	45
Figura 9a Esquema transformación SPEM-BPMN	49
Figura 9b Esquema transformación SPEM-BPMN	50
Figura 10 Metamodelo de Recursos Humanos	50
Figura 11 Diagrama de actividad <i>Análisis de Requerimientos</i>	51
Figura 12 Diagrama del Proceso de Negocio <i>Análisis de Requerimientos</i>	58
Figura 13 Diagrama de la actividad <i>análisis de requerimientos</i> en EPF Composer	63
Figura 14 Diagrama BPMN de la actividad <i>análisis de requerimientos</i> obtenida con MOSKitt4ME	63
Figura 15 Diagrama BPMN de la actividad <i>análisis de requerimientos</i> obtenida con la transformación implementada	64
Figura 16 Diagrama de la actividad <i>planificación de proyecto</i> en EPF Composer	65
Figura 17 Diagrama de la actividad <i>planificación de proyecto</i> obtenida con MOSKitt4ME	66
Figura 18 Diagrama BPMN de la actividad <i>planificación de proyecto</i> obtenida con la transformación implementada	66
Figura 19 Diagrama de la actividad <i>acuerdo comercial</i> en EPF Composer	67
Figura 20 Diagrama BPMN de la actividad <i>acuerdo comercial</i> obtenida con MOSKitt4ME	68
Figura 21 Diagrama BPMN de la actividad <i>acuerdo comercial</i> obtenida con la transformación implementada	68
Figura 22 Diagrama de la actividad <i>lanzamiento</i> en EPF Composer .	69
Figura 23 Diagrama BPMN de la actividad <i>lanzamiento</i> obtenida con MOSKitt4ME	69

Figura 24	Diagrama BPMN de la actividad <i>lanzamiento</i> obtenida con la transformación implementada	69
Figura 25	Diagrama de la actividad <i>validación de arquitectura</i> en EPF Composer	70
Figura 26	Diagrama BPMN de la actividad <i>validación de arquitectura</i> obtenida con MOSKitt4ME	71
Figura 27	Diagrama BPMN de la actividad <i>validación de arquitectura</i> obtenida con la transformación implementada	71
Figura 28	Diagrama de la <i>control de cambios</i> en EPF Composer	72
Figura 29	Diagrama BPMN de la actividad <i>control de cambios</i> obtenida con MOSKitt4ME	73
Figura 30	Diagrama BPMN de la actividad <i>control de cambios</i> obtenida con la transformación implementada	74
Figura 31	Diagrama de <i>diseño de pruebas</i> en EPF Composer	75
Figura 32	Diagrama BPMN de <i>diseño de pruebas</i> obtenida con MOSKitt4ME	75
Figura 33	Diagrama BPMN de la actividad <i>diseño de pruebas</i> obtenida con la transformación implementada	75
Figura 34	Diagrama de la actividad <i>desarrollo de solución</i> en EPF Composer	76
Figura 35	Diagrama BPMN de la actividad <i>desarrollo de solución</i> obtenida con MOSKitt4ME	77
Figura 36	Diagrama BPMN de la actividad <i>desarrollo de solución</i> obtenida con la transformación implementada	77
Figura 37	Diagrama de la actividad <i>Ejecución de Pruebas</i> en EPF Composer	78
Figura 38	Diagrama BPMN de la actividad <i>Ejecución de Pruebas</i> obtenida con MOSKitt4ME	78
Figura 39	Diagrama BPMN de la actividad <i>Ejecución de Pruebas</i> obtenida con la transformación implementada	79
Figura 40	Diagrama de la actividad <i>Liberación</i> en EPF Composer ..	79
Figura 41	Diagrama BPMN de la actividad <i>Liberación</i> obtenida con MOSKitt4ME	79
Figura 42	Diagrama BPMN de la actividad <i>Liberación</i> obtenida con la transformación implementada	79
Figura 43	Diagrama actividad <i>Implementación</i> en EPF Composer	80
Figura 44	Diagrama BPMN de la actividad <i>Implementación</i> obtenida con MOSKitt4ME	81
Figura 45	Diagrama BPMN de la actividad <i>Implementación</i> obtenida con la transformación implementada	81
Figura 46	Comparación de Diagramas BPMN obtenidos	86
Figura 47	Diagrama BPMN actividad <i>Desarrollo Solución</i> con pool y lanes	88
Figura 48	Escenarios de asignación de tareas con dos desarrolladores	89

Lista de tablas

	<u>Pág.</u>
Tabla 1 Estereotipos incluidos en el perfil UML de SPEM	30
Tabla 2 Estándares soportados en EPF Composer	34
Tabla 3 Versiones de UMA incluidas en EPF Composer	34
Tabla 4 Objetos de flujo en BPMN	36
Tabla 5 Conectores en BPMN	37
Tabla 6 Swinlanes en BPMN	38
Tabla 7 Artefactos en BPMN	38
Tabla 8 Mapeo entidades entre SPEM y BPMN	53
Tabla 9 Relaciones implementadas en la Transformación	54

Lista de Símbolos y abreviaturas

Abreviaturas

Abreviatura	Término
<i>OMG</i>	Object Management Group
<i>SPEM</i>	Software Process Engineering Metamodel
<i>UMA</i>	Unified Metamodel Architecture
<i>EPF Composer</i>	Eclipse Process Framework Composer
<i>BPM</i>	Business Process Management
<i>BPMS</i>	Business Process Management System
<i>BPMN</i>	Business Process Model and Notation
<i>UML</i>	Unified Modeling Language
<i>XSLT</i>	Extensible Stylesheet Language Transformations
<i>BPEL</i>	Business Process Execution Language
<i>XPDL</i>	XML Process Definition Language

Capítulo 1. Introducción

La presente tesis se enmarca el ámbito de la Ingeniería Dirigida por Modelos (MDE, *Model Driven Engineering*) y aborda dos retos principales: una transformación que permita obtener de manera semiautomática diagramas de procesos de negocio (en BPMN) a partir de procesos de Desarrollo de Software (en SPEM) y la definición de un metamodelo para la contextualización de recursos humanos.

La transformación teniendo como metamodelo objetivo a BPMN permitirá la aproximación a plataformas específicas más cercanas a modelos ejecutables como XPDL o BPEL, modelos fuente para BPMS. De otro lado, el metamodelo de recursos humanos, se presenta como una alternativa para instanciar a los participantes de un proyecto de desarrollo de software con una especificidad mayor a la que permite expresar el metamodelo SPEM. Este aumento de especificidad permitirá representar información significativa para la asignación de tareas a recursos humanos dentro de los proyectos.

En las secciones posteriores se presentan los conceptos claves que permitirán la contextualización del problema de investigación planteado, la solución propuesta y los objetivos del trabajo.

1.1 Proceso de Desarrollo de Software

Un proceso de desarrollo de software es un conjunto parcialmente ordenado de tareas, roles, productos y recursos de trabajo asociados para obtener un producto de software [1]. Las empresas de desarrollo de software definen procesos de desarrollo organizacionales con el objeto de mejorar su productividad y desempeño. Un proceso de desarrollo organizacional debe ser continuamente adaptado / instanciado a las necesidades específicas de cada proyecto que realice la empresa, a partir de sus diferencias en términos del tamaño de su equipo, la calificación del personal, conocimiento del dominio, y el riesgo asociado, entre otros factores.

Un metamodelo es un modelo a partir del cual se define la estructura, la semántica y las restricciones para una familia de modelos. SPEM (Software Process Engineering Metamodel) es un metamodelo especificado por el consorcio OMG (Object Management

Group)[2], usado para definir un proceso de desarrollo de software o una familia de procesos de desarrollo de software relacionados. La especificación SPEM está estructurada como un perfil UML (Unified Modeling Language) y como un metamodelo completo basado en MOF (Meta Object Facility).

El metamodelo SPEM está diseñado sobre el concepto de que un proceso de desarrollo de software es una colaboración entre entidades activas abstractas llamadas roles del proceso, que realizan operaciones llamadas tareas en entidades concretas y tangibles llamadas productos de trabajo. Los roles no corresponden a personas particulares, sino a perfiles de personas.

1.2 Gestión de Procesos de Negocio BPM

Un proceso de negocio es un conjunto estructurado de actividades lógicamente relacionadas, que tomando una o varias entradas, produce un resultado que tiene valor para una organización, en la medida que contribuye a alcanzar sus objetivos [3]. Un proceso de desarrollo de software es un proceso de negocio específico para empresas de la industria del software.

BPM (Business Process Management) se define como la habilidad de gestionar las actividades que son parte del ciclo de vida de un proceso de negocio. Esto implica descubrir, diseñar, desplegar, ejecutar, interactuar, operar, optimizar y analizar procesos desde la perspectiva de diseño de negocio, no de su implementación técnica [4].

Un BPMS (Business Process Management System) es una plataforma TI construida para gestionar procesos de negocio. Un BPMS desplegado en una organización será el responsable de coordinar las transacciones definidas por el proceso, manejar las instancias de los procesos, y procesar las transacciones distribuidas [4]. Un BPMS extiende la tecnología de workflow que predominó en los años noventa, adicionando mayor control de las conversaciones entre las entidades, control y gestión de diferentes hilos de ejecución, ejecución paralela, manejo de errores, compensación de transacciones y soporte para datos XML complejos, entre otros.

Las fases generales para la implantación de un BPMS en una organización incluyen la definición de los procesos de negocio, normalmente mediante una notación formal que da origen a su correspondiente modelo, la configuración de los procesos, la ejecución y/o simulación de los mismos, y el control y análisis de las distintas ejecuciones.

Para el diseño de los procesos de negocio existen diferentes notaciones estándares. Dos de estas notaciones son BPMN (Business Process Model and Notation) [5] y UML AD (Unified Modeling Language, Activity Diagrams) [6]. BPMN y UML AD proporcionan una notación gráfica fácilmente legible para el modelado de los procesos. Sin embargo, BPMN permite mayor expresividad frente UML AD en la representación de patrones de datos, de control de flujo y de recursos [29].

BPMN permite un enfoque orientado a procesos para modelos de sistemas. De acuerdo a su documento de especificación [5], su objetivo es proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas y los desarrolladores técnicos, hasta aquellos que monitorizarán y gestionarán los procesos.

De acuerdo con la OMG, BPMN se centra en los procesos de negocio y UML AD se centra en el diseño de software y por tanto no son competidoras, sino diferentes puntos de vista sobre un sistema[7].

Para la ejecución de los procesos de negocio, es necesario realizar una traducción de la notación del diseño del proceso a un lenguaje de ejecución. Para esto la WfMC (Workflow Management Coalition), ha definido XPDL (XML Process Definition Language) [8] que especifica la definición y la importación/exportación de procesos. Su objetivo es permitir modelar procesos en una aplicación, de tal modo que este modelo pueda ser usado por otras aplicaciones de modelado y/o por otras aplicaciones que trabajen en el entorno de ejecución. OASIS (Advancing Open Standards for the Information Society) ha definido BPEL (Business Process Execution Language) [9] que consiste en un lenguaje de ejecución basado en XML diseñado especialmente para el control centralizado de la invocación de diferentes servicios Web. BPEL puede complementarse con XPDL cuando se requiera un nivel de orquestación de servicios complejo [39].

1.3 Arquitectura Dirigida por Modelos

MDA (Model-Driven Architecture) [10] ha sido definida por OMG con el objetivo de atender los retos inherentes a la integración de aplicaciones y los continuos cambios tecnológicos. Propone separar la especificación de la funcionalidad del sistema de su implementación sobre una plataforma concreta, con lo que se promueve la portabilidad, la interoperabilidad, y la reusabilidad de los sistemas.

Desde la perspectiva de MDA, son posibles transformaciones desde un modelo fuente a un modelo objetivo. Por ejemplo, el código de una aplicación se puede generar a partir de un modelo PIM

(Platform Independent Model), mediante sucesivas transformaciones de modelo. Para realizar una transformación entre modelos es necesario el uso de lenguajes de transformación para describir cada una de las definiciones de transformación que establecen las correspondencias entre modelo origen y modelo destino.

Existen diferentes herramientas y lenguajes que soportan MDA. Cada herramienta incorpora un mecanismo propio para el manejo de transformaciones. En el 2003 la universidad de Latvia inicia la definición de un lenguaje procedural basado en grafos, denominado MOLA (MModel transformation LAnguage) [11]. En octubre de 2005 se inicia el proyecto EPSILON, con el objeto de crear una plataforma para manipular modelos; en este marco definen el lenguaje ETL (Epsilon Transformation Language) que permite definir transformaciones con varios modelos de entrada para obtener varios modelos de salida [12]. El proyecto ATLAS (INRIA - Institut National de Recherche en Informatique et en Automatique & LINA Laboratoire d'Informatique de Nantes Atlantique) uno de los más representativos en este campo, definió un lenguaje de transformación de modelos y conjunto de herramientas ATL (ATLAS Transformation Language), en respuesta a un Request-For-Proposal lanzado por la OMG en el 2002. ATL es un lenguaje híbrido de transformaciones declarativas e imperativas. Posee un algoritmo de ejecución determinista, y trabaja junto con un compilador que traduce el código fuente a sentencias que son interpretadas por una máquina virtual construida por el proyecto [13]. En el 2008 OMG definió un lenguaje estándar para la transformación de modelos denominado QVT (Query / View / Transformation) [14]. QVT también tiene una naturaleza híbrida (declarativa/imperativa). Su especificación define tres paquetes principales, uno por cada lenguaje definido: QVTCore, QVTRelation y QVTOperational.

1.4 Problema a Resolver

Un Proceso de Desarrollo de Software permite planificar proyectos para la construcción de productos de software, facilitando el manejo del alcance, tiempo, recursos, riesgos y calidad del producto. La definición de un proceso de desarrollo de software es esencial para lograr el éxito en los proyectos de una empresa de desarrollo.

La definición y especificación formal de procesos de desarrollo de software es una actividad compleja, que resulta más fácil al hacer uso de alguna herramienta de software. En el mercado se encuentran, entre otras, las siguientes herramientas para ayudar en la definición y especificación de procesos de desarrollo de software: Eclipse Process Framework Composer¹, Objecteering

¹<http://www.eclipse.org/epf/>

Modeler², MagicDraw³ UML (SPEM plug-in), Rational Method Composer⁴ y Enterprise Architect⁵.

Se destaca la herramienta Eclipse Process Framework Composer, -en corto, EPF Composer- que integra componentes para la creación, configuración, visualización y publicación de métodos y procesos de desarrollo de software. Internamente usa el metamodelo UMA (Unified Method Architecture), basado en el metamodelo SPEM. Permite además el intercambio de diagramas especificados con el perfil UML bajo el metamodelo XMI (XML Metadata Interchange). Estas características, además del licenciamiento libre, han hecho que EPF Composer sea una herramienta de amplio uso en las PyMEs de software[40].

La complejidad inherente a la administración de un proceso de desarrollo de software supone un amplio conjunto de conocimientos, habilidades y actividades para el equipo gestor del proceso, lo cual hace que esta labor además de crítica, represente altos costos para la empresa. La definición y especificación formal de un proceso de desarrollo de software, si bien favorece la administración de los proyectos de software, no permite por sí misma la gestión automatizada de los proyectos.

Los proyectos de desarrollo de software son instancias de procesos de negocio para la industria del software. La gestión automatizada de estos proyectos se consigue con la implementación al interior de la organización de una plataforma BPMS. Este es el responsable de coordinar las actividades definidas por el proceso, manejar las instancias y procesar las transacciones distribuidas.

Para gestionar automáticamente un proyecto de desarrollo de software a través de un BPMS, se requiere la transformación del proceso de desarrollo a proceso de negocio. Un proceso de negocio se modela conforme a BPMN y puede también transformarse a modelos conforme a BPEL o XPD. L.

BPMN, en su documento de especificación [5], establece explícitamente que el modelado de estructuras organizativas y de recursos queda fuera de su propósito. Sin embargo, incluye conceptos como *pool* y *lane* para expresar participantes y roles en un proceso de negocio. Un *pool* representa a un participante de un proceso, además actúa como un contenedor gráfico para particionar un conjunto de actividades desde otros *pools*. Un *lane* es una sub-partición dentro de un *pool* y extiende la longitud del *pool*, verticalmente u horizontalmente. Los *lanes* se usan para organizar y categorizar actividades.

² <http://www.objectteering.com/>

³ <http://www.nomagic.com/products/magicdraw-no-cost-add-ons/spem-plugin.html>

⁴ <http://www-03.ibm.com/software/products/us/en/rmc/>

⁵ <http://www.sparxsystems.com/products/ea/>

Las empresas que tienen sus procesos de desarrollo de software especificados en EPF Composer pueden usarlos como punto de partida hacia una transformación a procesos de negocio. Con los procesos de negocio obtenidos y el uso de un BPMS las empresas pueden alcanzar la gestión automática de sus proyectos. Sin embargo, un mismo proyecto de software no será gestionado en la misma forma independientemente del equipo humano disponible para su ejecución. Por lo tanto existen diferentes representaciones BPMN para un mismo proceso de desarrollo dependiendo del equipo humano con el que se cuente para su ejecución.

El problema a resolver es obtener la representación más apropiada para gestionar un proyecto que siga el proceso de desarrollo de software de una organización utilizando BPMN. Tal representación debe ser consistente con el proceso de desarrollo organizacional definido y también debe ser consistente con el equipo humano disponible para la ejecución del proyecto.

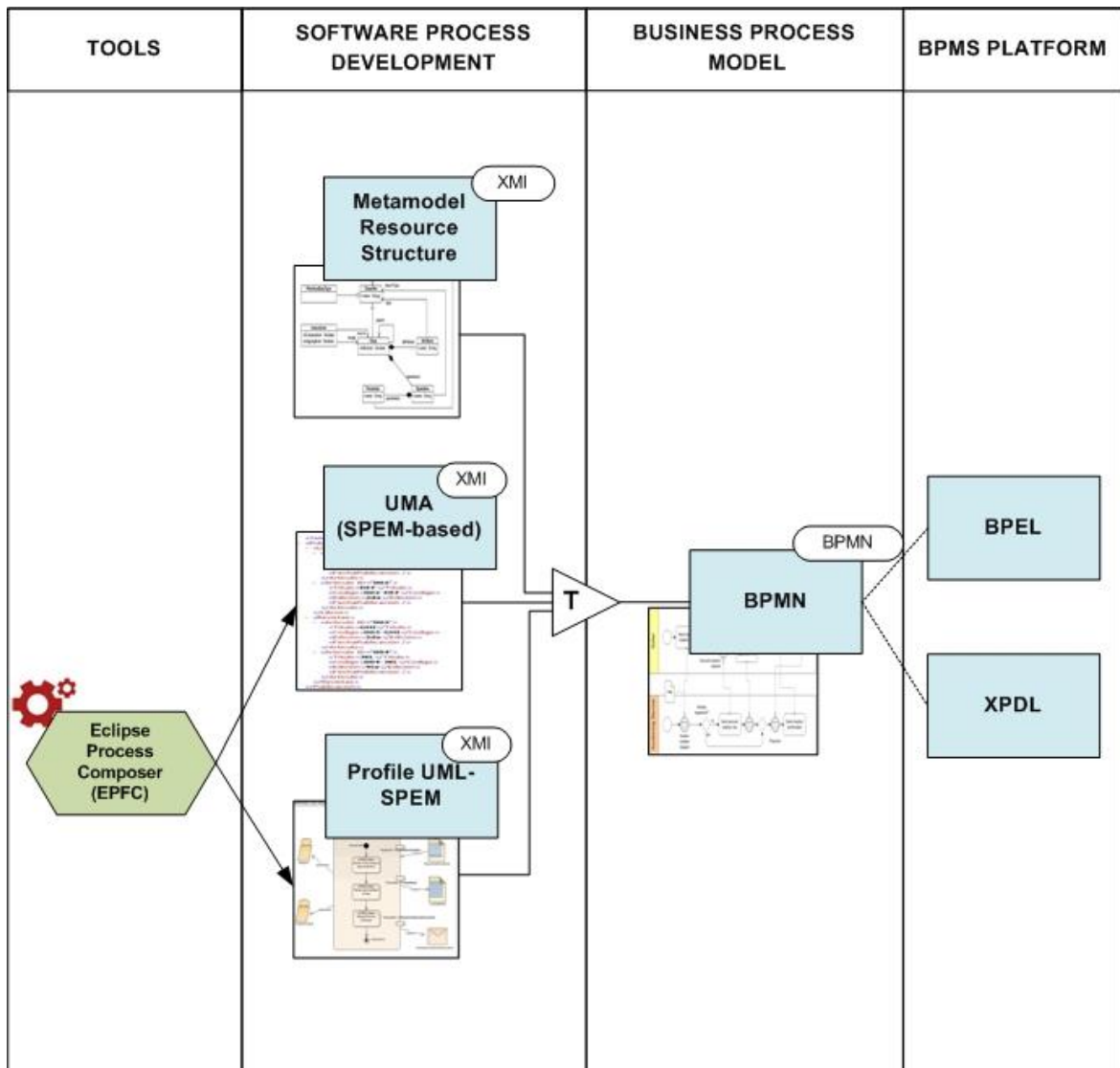
1.5 Solución Propuesta

Al realizar una transformación del *proceso de desarrollo de software* especificado en SPEM a un *proceso de negocio* modelado en BPMN se podría lograr la gestión automatizada de los proyectos que se ejecutan al interior de la empresa, con beneficios tales como mayor control para la gestión de las fases del proyecto, asistencia en su ejecución, mayor visibilidad sobre los ciclos y extracción automatizada de métricas.

La solución del problema implica la especificación de un metamodelo de recursos humanos y su instanciación para el equipo humano disponible para cada proyecto.

La transformación debe consumir dos modelos distintos derivados del *proceso de desarrollo de software* y que son generados por la herramienta EPF Composer. El primer modelo es *diagram.xmi*, que contiene la información referente al diagrama de actividad del proceso de desarrollo de software. El *diagram.xmi* está definido a través de los estereotipos del perfil UML de SPEM. El segundo modelo, *model.xmi*, contiene información relativa al *content method* del proceso de desarrollo y se especifica en UMA/SPEM. Estos dos modelos se compondrán con el *modelo de recursos humanos* y serán el origen de la transformación cuyo resultado es un *modelo conforme con BPMN* que contiene la representación notacional del *proceso de negocio* como se ilustra en la imagen (Figura 1).

Figura 1 Esquema de la solución planteada



1.6 Objetivos de la tesis

1.6.1 Objetivo General

Generar un framework para la transformación de procesos de desarrollo de software especificados bajo el metamodelo SPEM/UMA a procesos de negocio bajo un metamodelo BPMN que permita la contextualización de estructuras de recursos humanos.

1.6.2 Objetivos Específicos

- Definir un metamodelo para la especificación de estructuras de recursos humanos en procesos de desarrollo de software.
- Definir una transformación que tome como entrada un proceso conforme con el metamodelo UMA y un modelo de recursos humanos conforme con el metamodelo definido en el punto anterior, y produzca como salida un proceso conforme con el metamodelo BPMN y que incluya los recursos humanos concretos que participarán del proyecto.
- Establecer las herramientas computacionales que soporten las transformaciones entre modelos.
- Validar la estrategia de transformación a modelos BPMN en al menos dos proyectos de software de una PyME.

1.6.3 Resultados Esperados

- Metamodelo para la especificación de estructuras de recursos humanos en proyectos de desarrollo de software.
- Conjunto de reglas de transformación entre el metamodelo SPEM/UMA y el de recursos humanos como entrada, y el metamodelo BPMN como salida
- Guía de uso de los elementos construidos

1.7 Estructura de la Tesis

Este documento está organizado de la siguiente manera. En el primer capítulo se entregan algunos conceptos generales suficientes para permitir la identificación y el planteamiento del problema a solucionar y los objetivos generales de la presente investigación. En el capítulo dos se detallan aspectos específicos sobre definición de procesos de software, metamodelos UMA, SPEM, BPMN, transformación de modelos y se describen trabajos relacionados. En el tercer capítulo, se ilustran los detalles del análisis, diseño e implementación del metamodelo de recursos humanos y de las transformaciones que permiten obtener modelos en BPMN a partir de modelos SPEM. En el capítulo cuatro se valida los productos implementados mediante un caso de estudio. En el capítulo cinco se discuten los resultados y se plantean las conclusiones.

Capítulo 2. Background & Trabajos Relacionados

Existen diferentes notaciones para especificar formalmente procesos de desarrollo de software. La necesidad de unificar todos estos conceptos y notaciones condujo al consorcio OMG (Object Management Group) a la creación del metamodelo SPEM (Software Process Engineering Metamodel) presentado en el año 2002. A partir de la versión 1.1 de SPEM, IBM inició el desarrollo del metamodelo UMA (Unified Method Architecture), cuyo principal aporte es la definición del esquema y los conceptos para la representación de métodos. Finalmente, en el año 2007 OMG decide incluir los conceptos de UMA en la versión 2.0 de SPEM [2] [38].

En organizaciones de la industria del software los procesos de desarrollo son instancias de procesos de negocio. Transformar estos procesos de desarrollo a procesos de negocio es una primer fase para la implantación de un BPMS, que permitiría la gestión automatizada de los procesos de desarrollo. El estándar notacional para la especificación de procesos de negocios es BPMN (Business Process Model and Notation), mientras que para establecer la transformación entre los modelos SPEM y BPMN se puede hacer uso de lenguajes como QVT, ATL y XSLT, entre otros.

El presente trabajo incluye el desarrollo de una transformación en el lenguaje XSLT, con modelos fuente en SPEM (generados por EPF Composer) a modelos en BPMN. Por lo tanto, en las siguientes secciones se expondrán los conceptos fundamentales de SPEM, su especificación integrada UMA y su editor EPF Composer, BPMN y XSLT. Además una revisión del estado del arte de trabajos relacionados.

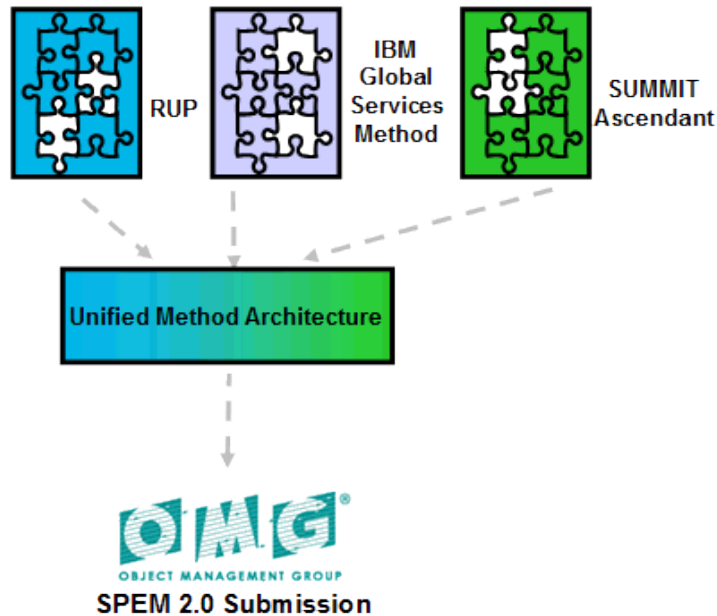
2.1 Unified Method Architecture (UMA)

La *arquitectura unificada de método* (Unified Method Architecture, UMA) es un metamodelo de ingeniería de procesos que permite la definición de esquemas y conceptos para la representación de métodos o procesos [34].

UMA se desarrolló para unificar de manera coherente diferentes métodos y lenguajes de ingeniería de procesos (ver Figura 2). UMA proporciona conceptos y posibilidades que tienen origen principalmente en los siguientes modelos:

- Extensión SPEM de UML para la ingeniería de procesos de software
- Lenguajes utilizados para RUP v2003
- Unified Process
- IBM Global Services Method
- IBM Rational Summit Ascendant.

Figura 2 Principales modelos que dan origen a UMA [35]



UMA establece los elementos clave para la representación de procesos en ingeniería de software. Esta representación incluye métodos, ciclos de vida, técnicas, roles, actividades, procesos, metodologías, plantillas, etc.

2.1.1 Principales aportes de UMA a SPEM

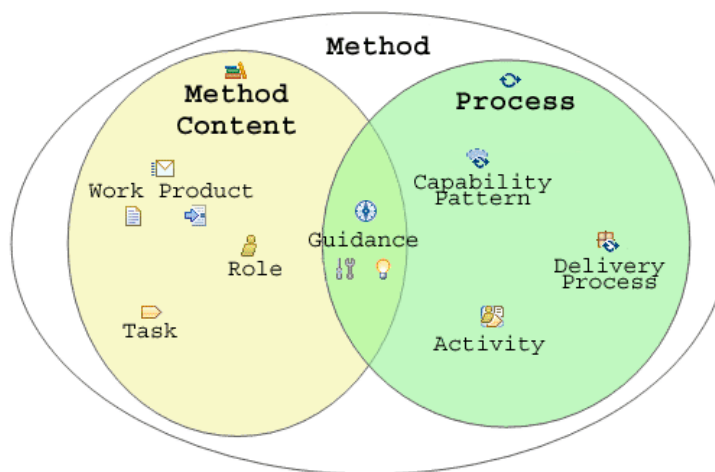
En el 2007 OMG adiciona al estándar SPEM la especificación UMA, de IBM. Esta inclusión le permitió a SPEM ampliar su capacidad para soportar las siguientes características [38].

Separación de contenido del método y proceso

UMA proporciona una clara separación de las definiciones del contenido del método de su aplicación en los procesos (Figura 3). Esto se obtiene a través de la definición por separado de los siguientes conceptos:

- **Method content** (contenido del método) soporta los conceptos que permiten construir una base de conocimiento de desarrollo reutilizable, en forma de descripciones de contenido general como roles, tareas, productos de trabajo y guías.
- **Method content in processes** combinación y reutilización de los elementos del *method content* para obtener aplicaciones específicas en forma de descripciones de procesos.

Figura 3 Elementos principales de UMA [36]



Reutilización del contenido

UMA permite que cada proceso haga referencia al *method content* común desde una agrupación de contenido del método común. A causa de estas referencias, los cambios en el contenido del método se reflejarán automáticamente en todos los procesos que lo utilicen. Sin embargo, la flexibilidad de UMA permite sobrescribir ciertos contenidos relacionados con el método en un proceso así como definir relaciones individuales específicas del proceso para cada elemento de proceso tales como añadir un producto de trabajo a una tarea, renombrar a un rol, o eliminar pasos de una tarea.

Familias del proceso

Además de soportar la representación de un proceso de desarrollo específico o el mantenimiento de varios procesos no relacionados, UMA proporciona un conjunto de conceptos para gestionar de forma coherente familias enteras de procesos relacionados. Para esto en UMA se definen los conceptos de *capability patterns* y *delivery processes* así como relaciones de reutilización específicas entre estos tipos de procesos. Estos conceptos permiten mantener familias coherentes de *delivery Process* que son específicos a un

tipo de proyecto definidos sobre variaciones de una del *method content* y los *capability patterns*.

UMA permite obtener distintas variantes de procesos específicos contruidos mediante la reutilización dinámica del mismo *method content* y los mismos *capability patterns*, pero aplicada con distintos niveles de detalle y de escala.

Diferentes ciclos de vida del proceso de software.

Como metamodelo de ingeniería de procesos, UMA proporciona soporte a diferentes modelos de ciclo de vida de desarrollo e inclusive a combinaciones de estos. Procesos gestionados a partir de ciclos como cascada, espiral, desarrollo iterativo, incremental, evolutivo, entre otros, pueden ser representados como modelos que conforman con el metamodelo UMA.

UMA define un conjunto de conceptos y atributos de adaptación para especificar la semántica temporal de elementos de proceso. Conceptos como fases, iteraciones, dependencias, trabajo continuo o condicionado por sucesos están incluidos en la especificación.

Extensibilidad y mecanismos de plug-in

UMA proporciona dos tipos de *plug-ins* que trabajan de diferentes maneras. Los *method plug-ins* proporcionan una forma de particularizar y adaptar el *method content*. Del mismo modo, proporciona los *process plug-ins* para obtener variaciones sobre *delivery Process* sin modificar su contenido original.

Múltiples vistas sobre procesos

UMA proporciona diferentes vistas para la definición de procesos. Estas vistas permiten a los ingenieros de procesos enfocar la definición del proceso basándose en sus preferencias personales. Un ingeniero de procesos puede optar por definir sus procesos centrándose en cualquiera de los aspectos siguientes:

- **Work Breakdown** (desglose de trabajo): vista centrada en el trabajo, desglosa las tareas asociadas a una actividad de mayor nivel.
- **Work Product Usage** (Uso del producto de trabajo): vista basada en resultados, define el estado de entregables y artefactos en distintos puntos del proceso.
- **Team Allocation** (Asignación de equipos): vista basada responsabilidades, define los roles necesarios y los productos de trabajo asociados.

UMA proporciona coherencia entre estas vistas, porque todas se basan en una estructura de objetos integrada. Los cambios en una vista se reflejan en las otras vistas.

2.2 Software Process Engineering Metamodel (SPEM)

SPEM es el estándar promulgado por OMG para la definición de procesos de desarrollo de sistemas y de software, así como para la definición y descripción de todos los elementos que los componen.

Para la versión 2.0 de SPEM, se plantearon los siguientes objetivos [2]:

- Asegurar la compatibilidad con UML 2.0.
- Definir un nuevo SPEM XML Schema que fuera compatible con MOF 2.0.
- Alinear SPEM con estándares emergentes que no sean UML, como BPMN.
- Permitir el desarrollo de extensiones para SPEM con el objetivo de que sean usadas por herramientas para conseguir la automatización de procesos.

Además, como se detalló en la sección anterior, la inclusión de UMA en la especificación SPEM 2.0 garantizó la cobertura de las siguientes características:

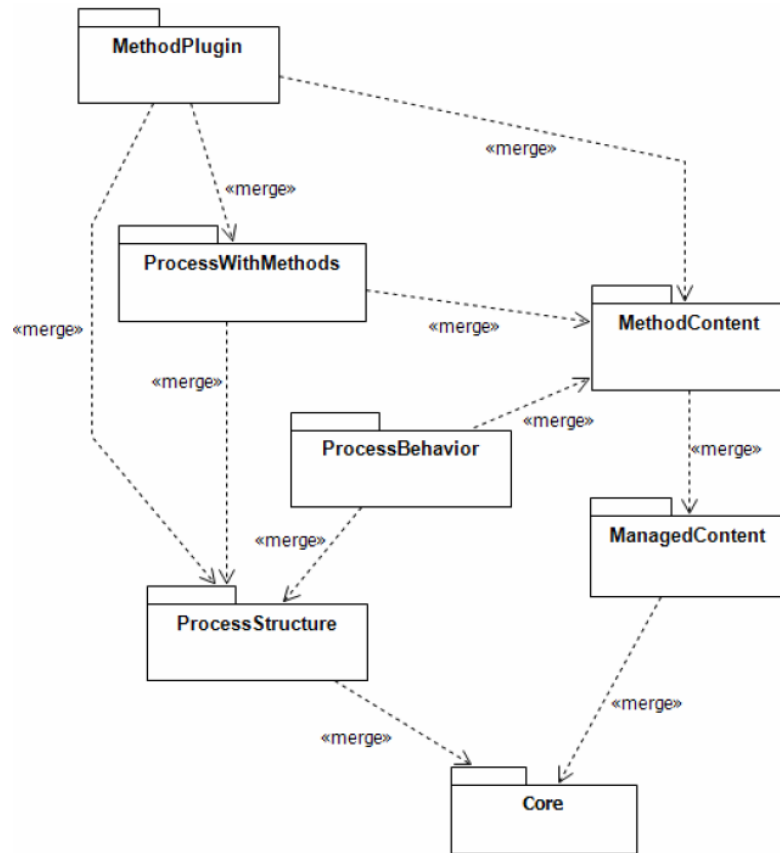
- Clara separación de la definición de los métodos de la aplicación de dichos métodos al desarrollo de un proceso concreto.
- Soporte para diferentes modelos de ciclo de vida.
- Mecanismo flexible para dar soporte a la variabilidad y extensibilidad de los procesos.
- Ensamblado rápido de procesos mediante el uso de patrones.
- Utilización de los principios de la encapsulación para conseguir componentes de proceso reemplazables y reusables.

2.2.1 Características principales de SPEM

Estructura del metamodelo SPEM

Con el propósito de permitir la organización y favorecer la reutilización, SPEM 2.0 está organizado en 7 paquetes. Cada paquete es una unidad lógica que extiende los paquetes de los que depende, proveyendo estructuras y capacidades adicionales. En la Figura 4, se muestran los paquetes y sus relaciones.

Figura 4 Paquetes del metamodelo de SPEM 2.0. [2]



Estos paquetes de SPEM proporcionan las siguientes capacidades:

- **Core:** Contiene todas las clases y abstracciones que constituyen la base para el resto de los paquetes del metamodelo.
- **Process Structure:** Contiene las clases necesarias para la creación de modelos de procesos.
- **Process Behavior:** Para representar el componente dinámico de los procesos; permite especificaciones sobre el comportamiento del proceso.
- **Managed Content:** Permite la incorporación y gestión de documentos, anotaciones y descripciones en lenguaje natural aplicada sobre información inherente a los procesos o sistemas de que no pueden ser expresadas como modelos.
- **Method Content:** Contiene los conceptos elementales de SPEM 2.0 que sirven de base para el ensamblado de procesos, metodologías y ciclos de vida. Los elementos de método permiten describir, con el detalle necesario, cómo se alcanzan los objetivos del proceso permitiendo la articulación de los roles con las tareas y relacionando el uso de los recursos con los resultados obtenidos. Estos conceptos son necesarios para construir la

base de conocimiento para el desarrollo que pueda ser utilizada independientemente del proceso o proyecto específico.

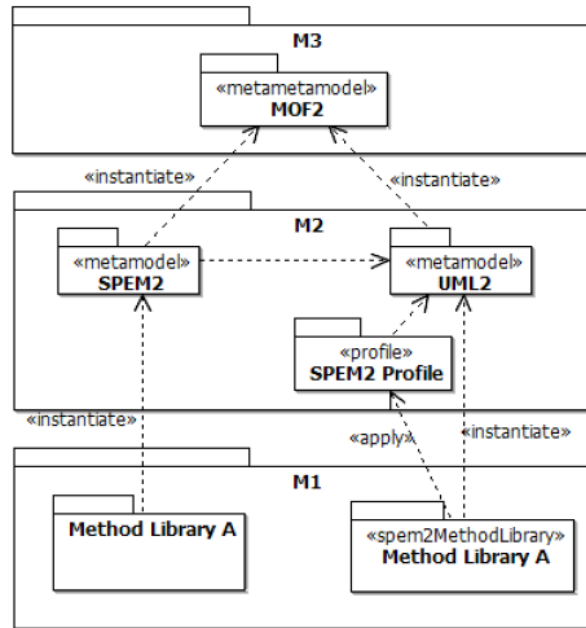
- **Process WithMethods:** Contiene los elementos necesarios para integrar los conceptos del paquete *Process Structure* con los conceptos y elementos del paquete *Content Method*.
- **Method Plug-in:** Introduce los conceptos para diseñar, gestionar y mantener repositorios y librerías de método y de procesos.

Perfil SPEM de UML 2

SPEM se describe de dos maneras. Por un lado se describe como *metamodelo* para definir todas las estructuras y reglas de estructuración para representar contenidos de métodos y procesos. Adicionalmente, se describe como un *perfil de UML*, es decir como un conjunto de estereotipos UML para la expresión de métodos y procesos a través del uso de las herramientas que han sido creadas inicialmente para trabajar con UML. En el primer caso (como metamodelo), SPEM se presenta como el conjunto de definiciones semánticas y restricciones; en el segundo caso, la definición sólo abarca la presentación.

La Figura 5, presenta la forma en que SPEM, el perfil de SPEM y UML se relacionan dentro de los niveles de abstracción de la OMG.

Figura 5 Niveles de modelado e Instanciación MDA















Aquí, la “Biblioteca de Métodos A” en el nivel M1 de la Figura 5 es un ejemplo de instancia concreta o modelo conforme al metamodelo SPEM como un esquema para representar su contenido. Esto implica que, mientras que SPEM define los conceptos de rol, tarea y producto de trabajo y las relaciones entre ellos, en la










“Biblioteca de Métodos A” se incluyen instancias concretas de dichos conceptos, tales como “analista de sistema” (instancia de Rol) y “Caso de Uso” (instancia de producto de trabajo).

Estereotipos del perfil UML de SPEM

Los estereotipos más usados del perfil, la Metaclase/Superclase de cada uno de ellos y el icono correspondiente para su representación gráfica se presentan en la Tabla 1. El conjunto de todos los estereotipos está ampliamente detallado en el documento de especificación de SPEM 2.0 [2].

Tabla 1 Estereotipos incluidos en el perfil UML de SPEM

Estereotipo	Meta/Superclase	Ícono
Activity	Work Definition, Planned Element /Action	
Category	Describable Element / Class	
CompositeRole	Role Use	
Guidance	Describable Element / Class	
Method Content Package	Package	
MethodLibrary	/Package	
MethodPlugin	/Package	
Metric	Guidance	
Milestone	WorkBreakdownElement / Classifier	
Process	Activity	
ProcessComponent	/Package	
ProcessPackage	Package	

Role Definition	MethodContentElement / Class	
RoleUse	BreakdownElement / Classifier	
Step	MethodContentElement, Work- Definition	
TaskDefinition	MethodContentElement, Work- Definition	
TaskUse	WorkBreakdownElement, PlannedElement / Classifier, Action	
TeamProfile	BreakDownElement / Classifier	
ToolDefinition	MethodContent Element / Class	
WorkProductDefinition	Method Content Element / Class	
WorkProductUse	BreakDownElement / Classifier	

2.3 Eclipse Process Framework Composer (EPF Composer)

EPF Composer es una herramienta del entorno Eclipse, desarrollado por la organización del mismo nombre, diseñada para crear, combinar y publicar métodos, procesos o metodologías de desarrollo de software de una organización. EPF Composer se alinea con el estándar SPEM 2.0 para sus implementaciones.

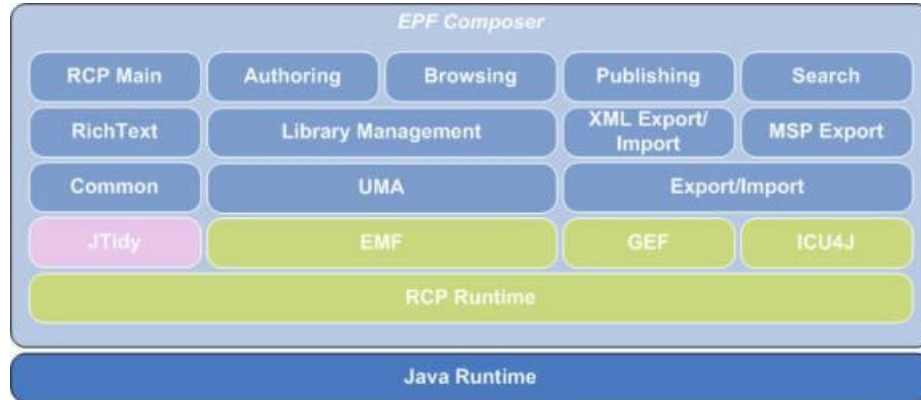
Desde su concepción, el proyecto EPF Composer se encaminó en la consecución de objetivos principales:

- Proporcionar un marco de trabajo extensible y una plataforma que facilite la ingeniería de procesos de software - creación de métodos y procesos, administración de bibliotecas, configuración y publicación de un proceso.
- Proporcionar contenido de proceso ejemplar y extensible para una gama de desarrollo y administración de procesos de software que soporte desarrollo iterativo, ágil e incremental, y sea aplicable a un amplio conjunto de plataformas y aplicaciones de desarrollo

2.3.1 Arquitectura de EPF Composer

EPF Composer es una aplicación java construida sobre una colección de componentes y estándares libres. La Figura 6 muestra la organización de los componentes claves en su arquitectura.

Figura 6 Arquitectura EPF Composer [33]



A continuación se presenta una descripción funcional para cada uno de los componentes que conforman la arquitectura de EPF Composer de acuerdo con lo especificado en [33].

- **Eclipse Rich Client Platform (RCP)**: Plataforma extensible para construir aplicaciones Java que como aplicaciones nativas de escritorio.
- **Eclipse Modeling Framework (EMF)**: Marco de trabajo de modelado y generación de código para construir aplicaciones con base en modelos de datos estructurados. EPF composer usa EMF para generar el dominio subyacente y las clases para acceder a la *method library*, importar y exportar el *method content* desde y hacia XML y exportar el contenido de la librería a Microsoft Project.
- **Graphical Editing Framework (GEF)**: Marco de trabajo para desarrollar editores gráficos y diagramas. Permite la visualización y edición de diagramas de actividades.
- **International Components for Unicode for Java (ICU4J)**: Librería para el soporte de *unicode* y de la internacionalización de software. EPF Composer lo usa para soportar la creación y publicación del *method content* y de procesos en múltiples lenguajes y lugares.
- **JTidy**: Inspector de sintaxis e impresora de HTML. EPF Composer lo usa para validar y formatear las representaciones en código XHTML de los elementos del método.
- **Common**: Provee servicios comunes de infraestructura a todos los componentes de EPF Composer. Los servicios principales incluyen *logging*, manejo de errores, manipulación de cadenas de

- caracteres, entrada y salida de archivos, análisis de XML, procesamiento de XSLT y formateo de HTML.
- **Unified Method Architecture (UMA):** Permite soporte al acceso y la edición de los elementos del método y los procesos almacenados en una *method library*. UMA define el metamodelo para la estructura del *method content* y los procesos en EPF.
 - **Library Management:** Soporta la interfaz de usuario y los servicios para administrar la *method library*. Entre los servicios principales se encuentran:
 - Crear una nueva *method library*, un *method plug-ins* y una nueva *method configurations*.
 - Abrir, guardar y hacer copia de una *method library* existente.
 - Persistir una *method library* a través de múltiples archivos XMI.
 - Administrar el proyecto de la *method library* en un espacio de trabajo.
 - Mostrar y filtrar el contenido de la *method library*.
 - Proveer comandos para editar el contenido de la *method library*.
 - Administrar las *method configurations*.
 - Administrar versiones.
 - Integración con sistemas de control de código tales como CVS y ClearCase
 - **RichText:** Facilita capacidades de edición de texto enriquecido en los editores de elementos del método. Es implementado usando una combinación de un navegador SWT y DHTML.
 - **Authoring:** Este componente provee las vistas y los editores que constituyen la perspectiva *authoring*. Soporta la edición y creación de los elementos de la *method library*. Proporciona asistentes que guían al usuario en la creación de una nueva *method library*, un *plug-in* y una *method configurations*.
 - **Browsing:** Provee las vistas que constituyen la perspectiva *browsing* en EPF Composer. Esta perspectiva permite a los usuarios la búsqueda del contenido de una *method configurations* realizada en formato HTML.
 - **Publishing:** Permite la interfaz de usuario y los servicios para publicar una configuración o proceso a un sitio Web estático. El proceso de publicación incluye los siguientes pasos:
 - **Search:** Este componente provee la interfaz de usuario y los servicios para buscar elementos específicos del método en la *method library*.
 - **Export/Import:** Interfaz de usuario servicios para exportar e importar *method plug-ins* y *method configurations* empaquetados en archivos XMI.
 - **XML Export/Import:** Interfaz de usuario y servicios para exportar e importar *method plug-ins* y *method configurations* empaquetados en archivos XML.

- **Microsoft Project (MSP) Export:** Permite exportar un proceso a un archivo XML de Microsoft Project 2003. La estructura y el contenido de los archivos XML exportados están gobernados por el esquema Microsoft Office Project2003 XML.
- **RCP Main:** Este componente provee personalización específica del producto que define la apariencia, el empaquetado y el sitio de actualización de software en EPF Composer. También contiene la pantalla de bienvenida del EPF Composer y contenido tal como la descripción, ejemplos y tutoriales.

2.3.2 Estándares usados en EPF Composer

El más reciente *release* de EPF Composer, liberado en 2012 como 1.5.1.6 se hace uso de los estándares detallados en la **Tabla 2** [33].

Tabla 2 Estándares soportados en EPF Composer

XMI (<i>XML Meta Interchange</i>)	Especificación de OMG [31] para el almacenamiento e intercambio de metadatos en formato XML.
UML 2.0 -Diagram Interchange specification	Especificación de OMG para el intercambio de diagramas de UML entre diferentes herramientas de modelado.
XSLT <i>Extensible Stylesheet Language Transformation</i>	Recomendación de World Wide Web Consortium (W3C) para la transformación de documentos XML en otras formas de documentos XML.
DHTML <i>Dynamic HTML</i>	Comprende especificaciones como HTML, JavaScript, Cascading Style Sheets (CSS) y Document Object Model (DOM).

2.3.3 Relación de versiones de EPF Composer & UMA

La Tabla 3 relaciona las versiones de EPF Composer con la versión de UMA que utiliza. En el desarrollo de la presente investigación se hizo uso de la versión 1.5 de EPF Composer, que incluye la versión 1.0.5 de UMA [32].

Tabla 3 Versiones de UMA incluidas en EPF Composer

EPF Composer versión	UMA versión	SPEM versión
Beta M3	1.0.2	
1.0		
1.0.1	1.0.3	

1.0.2		
1.2	1.0.4	2.0
1.5	1.0.5	2.0

2.4 Business Process Model and Notation (BPMN)

BPMN (Business Process Model and Notation), es un estándar de la BPMI (Business Process Management Initiative), organismo que ha sido acogido por OMG, cuyo principal objetivo es, según su documento de especificación [5], "proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas... los desarrolladores técnicos... hasta aquellos que monitorizarán y gestionarán los procesos". Otros objetivos importantes que se plantea esta especificación, incluidos en el mismo documento son:

- Crear puentes entre el diseño de los procesos de negocio y la implementación del proceso.
- Proporcionar una notación gráfica a los lenguajes basados en XML para describir procesos (como BPEL4WS).

BPMN incluye conceptos y recoge experiencias de diferentes estándares, principalmente:

- Diagramas de Actividad de UML
- UML Profile for Enterprise Distributed Object Computing (UML EDOC)
- Integrated DEFinition Methods (IDEF)
- Electronic Business using eXtensible Markup Language (ebXML)
- Activity-Decision Flow (ADF Diagram)
- RossetaNet
- Line of Visibility Engineering Methodology (LOVeM)
- Event-driven process chain (EPC)

2.4.1 Diagramas BPMN

BPMN provee una notación gráfica que permite la creación de variedad de diagramas dentro de los tres tipos de submodelos:

- Procesos de negocio privados (internos).
- Procesos abstractos (públicos).
- Procesos de colaboración (globales).

Permite también la creación de diagramas con una combinación de tales tipos de modelos. Sin embargo en su especificación se advierte que "debemos tener cuidado si combinamos demasiados tipos de submodelos... obtendremos un diagrama difícil de entender...

por eso se recomienda al modelador que se centre en un tipo de modelo para los diagramas”.[5]

Los diagramas en BPMN, están formados por un conjunto de elementos fundamentales. Estos se clasifican en cuatro categorías [5]:


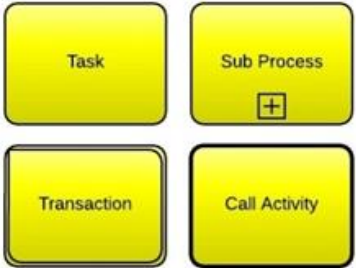
- Flow objects (Objetos de Flujo)
- Connecting Objects (Conectores)
- Swimlanes (Calles)
- Artifacts (Artefactos)

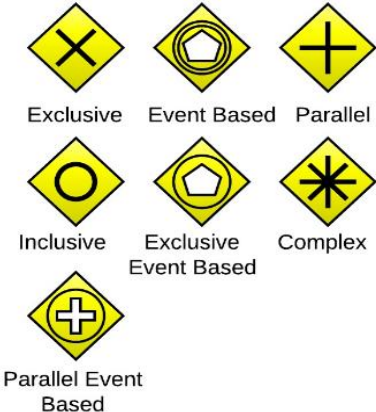
En las siguientes tablas se describe con mayor detalle cada uno de los elementos que conforman el estándar. Se presentan agrupados de acuerdo a la anterior categorización y se incluye su ícono asociado.

Flow objects

Elementos principales descritos dentro de BPMN que definen el comportamiento de los procesos.

Tabla 4 Objetos de flujo en BPMN




Tipo	Descripción	Ícono
Events (eventos)	Algo que ocurre durante el desarrollo de un proceso de negocio. Pueden ser de tres tipos: inicio, intermedio y finalización	
Activity (Actividades)	El término genérico para denominar cualquier trabajo que se realiza en la organización. Pueden ser atómicas o compuestas	

<p>Gateway (Pasarelas o Compuertas)</p>	<p>Para controlar el flujo, puede ser una decisión tradicional, un join, un merge o un fork</p>	 <p>Exclusive Event Based Parallel</p> <p>Inclusive Exclusive Event Based Complex</p> <p>Parallel Event Based</p>
--	---	--

Connecting Objects

Elementos para conectar los diferentes *Flow Objects*. Permiten definir el esqueleto estructural básico de procesos de negocio.

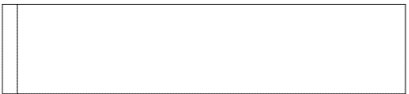

Tabla 5 Conectores en BPMN

Tipo	Descripción	Ícono
<p>Secuence Flow (Flujo de secuencia)</p>	<p>Para indicar el orden en el cual son ejecutadas las actividades del proceso de negocio</p>	
<p>Message Flow (Flujo de mensaje)</p>	<p>Para mostrar el intercambio de mensajes entre dos participantes (entidades de negocio o pools)</p>	
<p>Association (Asociación)</p>	<p>Para asociar artifacts con flow objects</p>	

Swinlanes

Swinlanes son el mecanismo que permite clasificar las actividades de manera visual. Permite ilustrar distintas categorías o responsabilidades



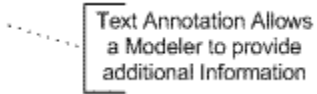
Tabla 6 Swinlanes en BPMN

Tipo	Descripción	Ícono
Pool	Para indicar los participantes en el proceso de negocio	
Lane	Es una partición (vertical u horizontal) del pool. Permite clasificar las actividades	

Artefactos

Los artefactos permiten adicionar mayor información al modelo o diagrama. De esta manera, el modelo se hace más legible. Son tres tipos de artefactos predefinidos, pudiéndose añadir artefactos adicionales.

Tabla 7 Artefactos en BPMN

Tipo	Descripción	Ícono
Datos (Data Object)	Representación de los datos que son producidos o requeridos por las actividades	
Grupo (Group)	Para agrupar distintos elementos del diagrama	
Anotaciones (Annotations)	Para proporcionar información de texto adicional	

2.4.2 Limitaciones de BPMN

BPMN es un estándar en desarrollo que, si bien ha alcanzado un nivel de madurez que le permite ser ampliamente usado en la industria, en su estado actual presenta ciertas limitaciones que

serán soportadas en futuras versiones [29]. BPMN no provee soporte para representar los siguientes modelos:

- Estructura de la organización
- Recursos humanos
- Modelos de datos
- Estrategias
- Reglas de negocio
- Flujo de datos, conforme el documento de especificación "aunque BPMN muestre el flujo de datos (mensajes) y las asociaciones de los artefactos con las actividades, no es un diagrama de flujo de datos".

2.5 Transformación de Modelos

En la literatura referente al ámbito de MDE, se encuentran diferentes definiciones para el concepto de *transformación de modelos*. A continuación, a modo de ejemplo se citan las más ampliamente usadas:

- Según la guía de MDA [10]: "La transformación de modelos es el proceso de convertir un modelo en otro modelo del mismo sistema.
- En 2003 en [30], se define una transformación de modelos como la generación automática de un modelo destino, partiendo de un modelo origen, conformes a la definición de la transformación.
- Por último, en [36] se define como la automatización de un proceso que tiene uno o más modelos origen y producen uno o más modelos destino como salida siguiendo un conjunto de reglas de transformación.

De acuerdo a la naturaleza (declarativa/imperativa) del lenguaje de transformación que se utilice y la estrategia de solución que se implemente, es posible identificar algunos de los siguientes componentes genéricos en una transformación:

- Metamodelos participantes.
- Conjunto de reglas que especifican la relación existente entre términos de los metamodelos.
- Conjunto de dominios por regla que se ajusta al conjunto de términos para los que se expresan.
- Relaciones, un conjunto de patrones que cumplen con la estructura de los términos.

Como se detalló en la sección 1.3, existen diferentes herramientas y lenguajes para la implementación de una transformación de modelos. Los lenguajes disponibles son entre otros MOLA, ETL, ATL y QVT.

QVT es el estándar de la OMG para la transformación de modelos. En su definición se han especificado tres lenguajes distintos QVTCore, QVTRelation y QVTOperational. Aun siendo el estándar, el uso de QVT es bastante limitado, debido a las siguientes dificultades:

- Soporta solamente transformaciones de modelo a modelo (uno a uno).
- No hay disponibilidad de librerías y herramientas suficientemente maduras, que proporcionen el entero soporte a las características del estándar.
- Por último, ya que cada herramienta posee una representación propia de los modelos, la interoperabilidad entre las herramientas es escasa.

Las dificultades enunciadas motivan la selección de lenguajes de transformación de modelos alternativos. Otro tipo de herramientas disponibles para la representación y transformación de modelos son las basadas en Extensible Markup Language (XML) [8]. Para la representación de modelos se encuentra el estándar XML Metadata Interchange (XMI) [31] de la OMG y para la especificación de las transformaciones es posible el uso de XSLT (Extensible Stylesheet Language Transformations) de W3C [41].

2.5.1 Lenguaje XSLT para transformación de modelos

XSLT es un lenguaje de programación declarativo que, usado desde la perspectiva MDE permite generar modelos a partir de modelos fuente siempre y cuando éstos se estructuren como XML.

Inicialmente el uso de XSLT se orientó principalmente a formatos de presentación y estilo como HTML, XHTML, o SVG. Sin embargo la evolución de sus posibilidades técnicas ha permitido el desarrollo de tareas de transformación más elaboradas.

XSLT 1.0 fue parte de la recomendación XLS del consorcio W3C publicada en 1999. En abril de 2009 se publicó la versión 2.0 de esta recomendación y en julio de 2012 se publicó un draft de la versión 3.0 con marcadas mejoras para la transformación de streaming (flujo de datos de transformaciones) y características que permitirían la modularidad en modelos complejos.

Una transformación en el lenguaje XSLT está expresada en forma de stylesheet, cuya sintaxis es conforme con XML y conforme a la recomendación Namespaces in XML [42].

Una transformación expresada en XSLT describe reglas para transformar un modelo de entrada en uno o más modelos de salida. XSLT implementa una estructura en forma de árbol para la representación de los modelos. La estructura de estos árboles está

definida en la recomendación de la W3C XQuery 1.0 and XPath 2.0 Data Model (XDM) [43]. La transformación se realiza mediante la aplicación de un conjunto de reglas. Una regla asocia un patrón que coincide con los nodos en el árbol fuente con un constructor de secuencia. Un constructor secuencia es una secuencia de cero o más nodos que se evalúa para devolver una secuencia de nodos y valores atómicos. XSLT permite especificar la forma en que se utiliza la secuencia resultante de la evaluación. En algunos casos, la evaluación originará la construcción de nuevos nodos, los cuales pueden usarse para producir parte del árbol resultante. La estructura de los árboles resultantes puede ser completamente diferente de la estructura de los árboles fuente.

2.6 Trabajos relacionados

En esta sección se muestra el estudio realizado sobre el estado del arte de los tópicos relacionados con la presente investigación. Se realizó una búsqueda estructurada para conocer las diferentes propuestas para la transformación de procesos de software a procesos ejecutables XPDL o BPEL y una búsqueda no estructurada para determinar las propuestas relacionadas con la definición de metamodelos de recursos humanos.

2.6.1 Transformación de Modelos de Procesos de Desarrollo de Software a Modelos de Procesos de Negocio

Los procesos de desarrollo de software especificados en SPEM son modelos estructurales de esquemas y conceptos. Los procesos en SPEM no pueden ser gestionados automáticamente dado que no son ejecutables a través de plataformas BPMS; para lograr la gestión automática se han planteado diferentes propuestas de transformación de modelos SPEM a notaciones como BPMN, XPDL y BPEL.

Con el propósito de determinar las estrategias, métodos, herramientas y lenguajes más adecuados para la especificación de transformaciones entre los modelos y cuál ha sido el beneficio de tales transformaciones cuando se aplica en escenarios reales, se realizó una búsqueda de trabajos relevantes. La búsqueda se aplicó sobre dos bases de datos: IEEE Xplore Digital Librar y ACM Digital. El estudio comprendió las siguientes cadenas de búsqueda:

- Transformation SPEM to PBEL
- Transformation SPEM to PBEL
- Transformation SPEM to BPMN
- Transformation SPEM to XPDL
- Transformation BPMN to PBEL
- Transformation BPMN to XPDL
- Transformation UML AD to PBEL

- Transformation UML AD to BPMN
- Transformation UML AD to XPDL

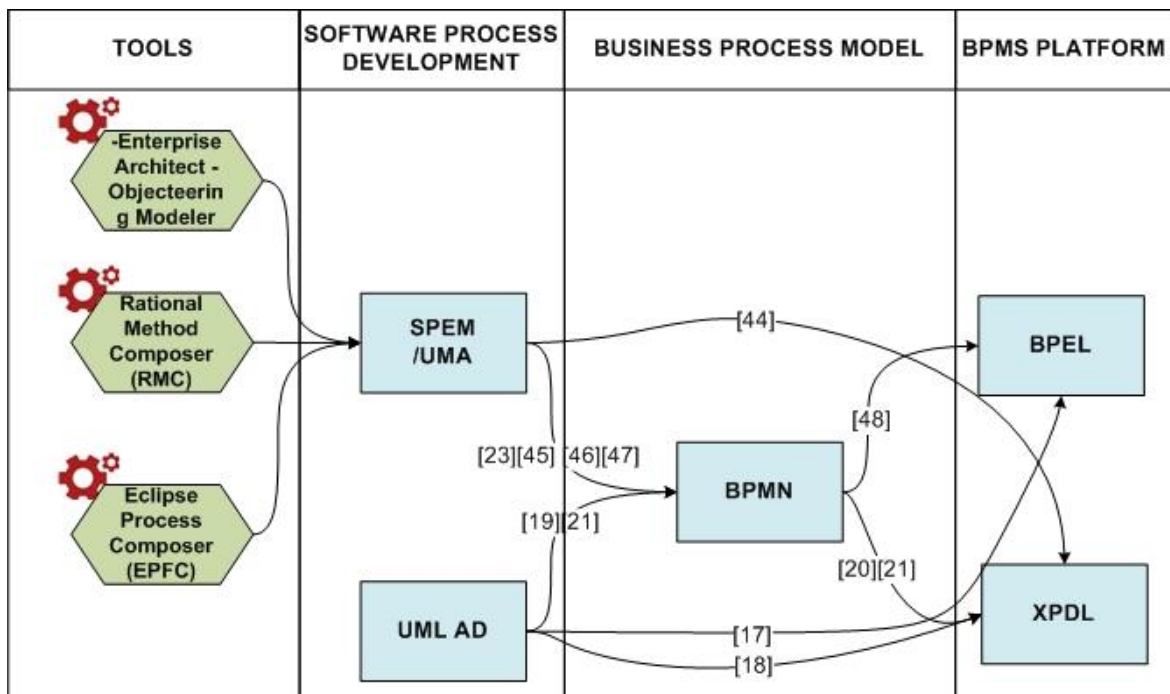
Se optó por incluir cadenas de búsqueda teniendo como modelo fuente a UML AD ya que SPEM es también un perfil de UML, con lo cual se amplía el espacio de búsqueda. En la tabla se presenta la relación del número de trabajos recuperados al aplicar la búsqueda sobre las bases de datos señaladas.

Objetivo Fuente	BPMN	BPEL	XPDL
SPEM	5 IEEE, 4 ACM	1 IEEE, 8 ACM	2 IEEE, 3 ACM
UML AD	8 IEEE, 11 ACM	5 IEEE, 9 ACM	1 IEEE, 2 ACM
BPMN		8 IEEE, 8 ACM	5 IEEE, 19 ACM

Eliminados los resultados que se repetían en dos o más categorías, se aplicaron diferentes filtros sobre los trabajos obtenidos: año de publicación, idioma y concordancia del resumen con el tema de interés.

Se encontró diferentes propuestas de transformaciones directas a procesos ejecutables XPDL o BPEL y transformaciones con modelos intermedios en BPMN. Finalmente los trabajos más relevantes se ilustran en la Figura 7. Los números en los conectores son la referencia a la bibliografía incluida en el presente libro.

Figura 7 Trabajos relevantes en la Transformación de Modelos



En 1997 en [15] se realiza la transformación del proceso de desarrollo ISPW-6 Software Process Example, en particular su fase de gestión de requisitos, a un lenguaje de especificación de workflow denominado Valmont. Aunque no es claro bajo que estándar está especificado el ISPW-6, logran la transformación de actividades, tareas, roles y las estructuras de control de flujo de ISPW-6 a el lenguaje ejecutable Valmont.

En 2003 en [16] se propone un entorno de trabajo basado en workflow denominado WSDPP. Este trabajo no entrega en si un enfoque de transformación de procesos a modelos ejecutables; sino un entorno para soportar gestión automática de proyectos de software. El entorno soporta la gestión de disciplinas como control de versiones, manejo de errores, gestión de cambio y listas de distribución. WSDPP es parametrizable, por lo cual se puede utilizar en diversos procesos de desarrollo, como los derivados de RUP, TSP, PSP y XP.

En 2003 en [17] se especifica un perfil UML consistente con los conceptos de BPEL para modelar procesos de negocio. Este trabajo permite obtener de manera automática un modelo ejecutable BPEL, desde un modelo en UML. En 2006 en [18] se definen reglas formales para la transformación de modelos UML AD a modelos XPDL 1.0. Las reglas de transformación definidas se implementan en un *plugin* para la herramienta MagicDraw. Se define también una optimización para los modelos XPDL obtenidos. En 2006 en [19] se presenta una transformación formal de UML AD a BPMN 1.0, utilizando el lenguaje MOLA. El trabajo está orientado a la especificación en BPMN de modelos de procesos sin considerar su ejecución.

En [44] se presenta una transformación de procesos especificados en SPEM a procesos en XPDL, con reglas de transformación implementadas en el lenguaje OCL (Object Constraint Language). La transformación se aplica al proceso de una empresa real con Shark como motor XPDL.

En [45] se propone una transformación de procesos de software en SPEM a proceso de negocio en BPMN usando QVT como lenguaje de transformación. El caso de estudio utilizado es una instancia particular de RUP (Rational Unified Process) definida para pequeños proyectos denominada SmallRUP. En [46] se formalizan las transformaciones de estructuras *breakdown elements* en SPEM a secuencias de actividades en BPMN, usando RSL (RAISE Specification Language) como lenguaje de especificación.

En 2007 en [20] se presenta una introducción a la herramienta bxModeller. Permite modelar procesos de negocio con BPMN 1.0 y traducir el diagrama BPMN a XPDL 2.0. Describe de manera informal las correspondencias que son implementadas en Java. En 2010, en [21] se aborda una transformación horizontal que permite la traducción de UML AD a BPMN 1.0 y una transformación desde BPMN a

XPDL. El trabajo incluye la definición de metamodelos parciales de UML AD, BPMN y XPDL, la implementación de cada una de las reglas de transformación con el uso de la herramienta SMART QVT[22], una implementación de código abierto para Eclipse que soporta el lenguaje QVT. Sin embargo este trabajo no presenta un caso de estudio aplicado.

En 2012 en [47] se presenta un detallado análisis comparativo entre los metamodelos SPEM y BPMN. Se discuten criterios como su expresividad, reusabilidad, administración, evolución, entendimiento e integración en la organización. Se incluye también una tabla en donde se presentan las correspondencias entre las entidades de los dos metamodelos. Éstas correspondencias fueron utilizadas para la especificación de las reglas de transformación.

En [48] se presenta ejemplo de cómo un diagrama de procesos en BPMN puede ser usado para representar un proceso ejecutable en BPEL. El trabajo no se desarrolla en el ámbito específico de los procesos de software. Se valida sobre un caso aplicado a un proceso de reserva de viaje.

En 2012 se presenta MOSKitt4ME [23] una herramienta cuyo desarrollo es liderado por la Universidad de Valencia en el marco del proyecto Modeling Software KIT (MOSKitt). MOSKitt tiene como objetivo ofrecer el soporte a la metodología gvMétrica (una adaptación de Métrica III) usada para sistematizar las actividades que dan soporte al ciclo de vida del software. MOSKitt tiene una arquitectura basada en *plugins* que la convierte no sólo en una herramienta CASE sino en toda una plataforma de modelado sobre software libre para la construcción de herramientas.

MOSKitt4ME es una herramienta basada en Eclipse. Integra dos componentes que permiten la especificación completa de un método o proceso. Por un lado incluye la herramienta EPF Composer, para la descripción de los fragmentos de métodos o procesos y por otro lado *Activity Designer* para la especificación del proceso.

MOSKitt4ME implementa una transformación parcial del proceso especificado en SPEM a modelos BPMN. En la versión actual simplemente se generan *tasks* y *sequence flows* a partir de las tareas y precedencias definidas en el proceso en SPEM. Para representar los demás elementos los autores sugieren realizar la edición manual del modelo BPMN a través de la herramienta *Activity Designer*, incluida en Moskitt.

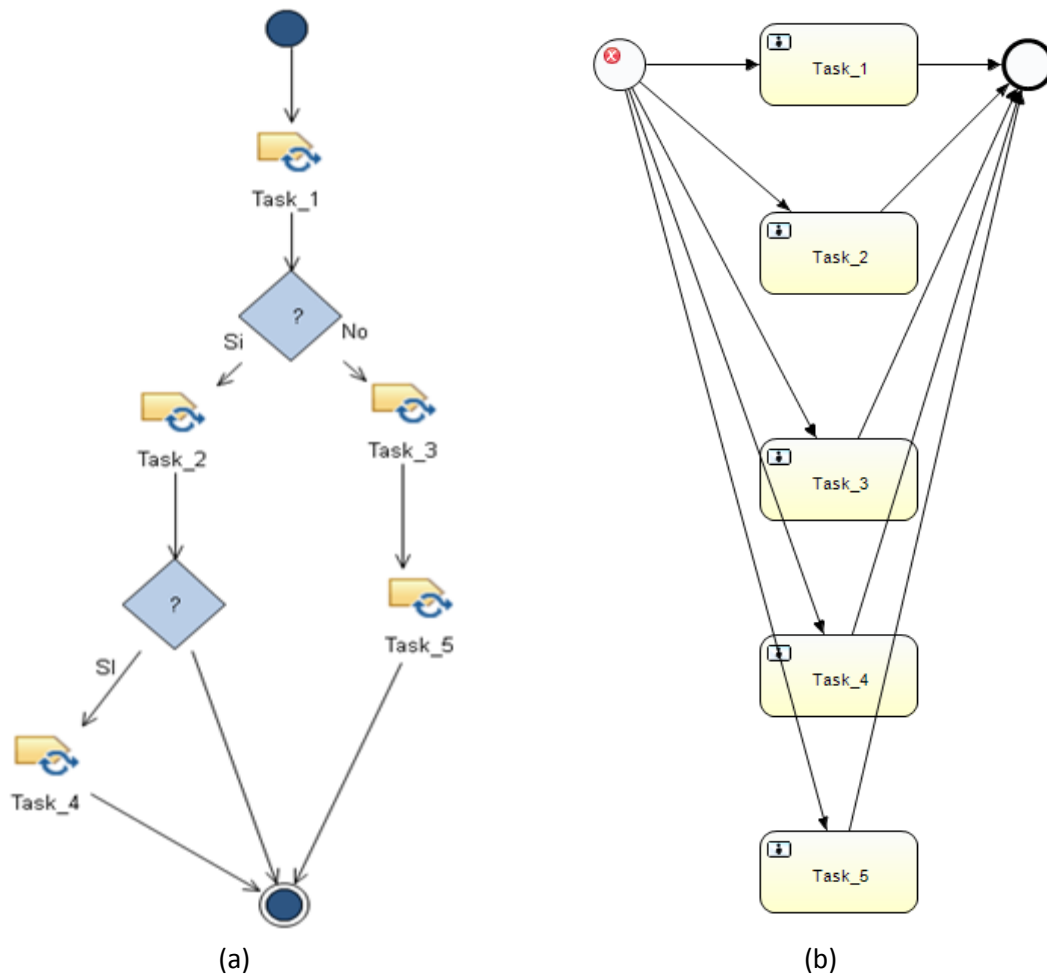
Otra limitante de MOSKitt4ME es que no representa los roles responsables de las tareas en el BPMN generados.

Para ilustrar las capacidades de MOSKitt4ME, se realizó la especificación de una actividad simple, compuesta por cinco tareas. El flujo de secuencia de la actividad incluye un nodo

inicio, nodo fin y dos nodos de decisión que se muestran en la Figura 8 (a).

El diagrama BPMN generado por la transformación automática que implementa MOSKitt4ME, es conforme con el número y nombre de las tareas del proceso en SPEM. Sin embargo, se evidencia que el BPMN no representa el flujo de secuencia original, se pierden las conexiones entre tareas y no aparecen los nodos de decisión como se muestra en la Figura 8 (b). Adicionalmente, en el diagrama BPMN no están expresados los roles responsables de la ejecución de cada tarea.

Figura 8 Transformación de SPEM a BPMN con MOSKitt4ME



2.6.2 Metamodelos de Recursos Humanos

Para establecer el estado del arte relativo a la representación de información de recursos humanos a través de metamodelos no fue posible aplicar una búsqueda estructurada, esto debido a que existen muy pocos trabajos en el área.

En [24] se define una extensión de BPMN 1.0 para expresar un conjunto de restricciones sobre los recursos humanos que participan en procesos de negocios.

En [25] se define un perfil UML basado en modelos de casos de uso denominado Human Task Profile. Permite la contextualización de estructuras de recursos y aspectos de la distribución del trabajo en una organización. En este trabajo se implementa una transformación desde el modelo definido en el perfil UML a un modelo conforme con el nuevo estándar WS-HumanTask (Web Service Human Task) [26]. Este estándar provee una notación, un diagrama de estado y una API para especificaciones de tareas desarrolladas por los recursos humanos en un proceso de negocio.

En 2011 en [27], se implementa una extensión al estándar BPMN 2.0 para la definición de recursos en tres sentidos: estructura de recursos, distribución de tareas y aspectos de autorización y definición de privilegios. Se valida la expresividad de la extensión obtenida sobre los Workflow Resource Pattern [28] como marco de evaluación, dando soporte a 38 de los 43 requerimientos de prueba allí propuestos.

Los escasos trabajos relacionados al modelamiento de recursos humanos terminan por especificar extensiones al estándar BPMN. Las extensiones pueden generar el detrimento de la portabilidad de los modelos .bpmn entre herramientas. En este trabajo se intentará mantener la consistencia frente al estándar para no tener este inconveniente.

Capítulo 3. Desarrollo de la Solución

EL objetivo principal del presente trabajo incluye la implementación de una transformación de modelos en SPEM (generados por EPF Composer) a modelos en BPMN.

Un esquema inicial propuesto para abordar la transformación se presenta en la Figura 9a. En esta se ilustran los dos modelos generados por la herramienta EPF Composer. El diagram.xml, que contiene la información referente al diagrama de actividad del proceso de desarrollo de software definido a través de los estereotipos del perfil UML de SPEM y el model.xml, con la información relativa al content method del proceso de desarrollo.

El desarrollo del proyecto evidenció inconveniente tal esquema de solución. Abordar transformaciones parciales en un mismo escenario, pero aplicadas una luego de la otra resolvió las serias dificultades que se tenían, por lo que el esquema final de trabajo fue modificado y se muestra en la Figura 9b

Figura 9a Esquema transformación SPEM-BPMN

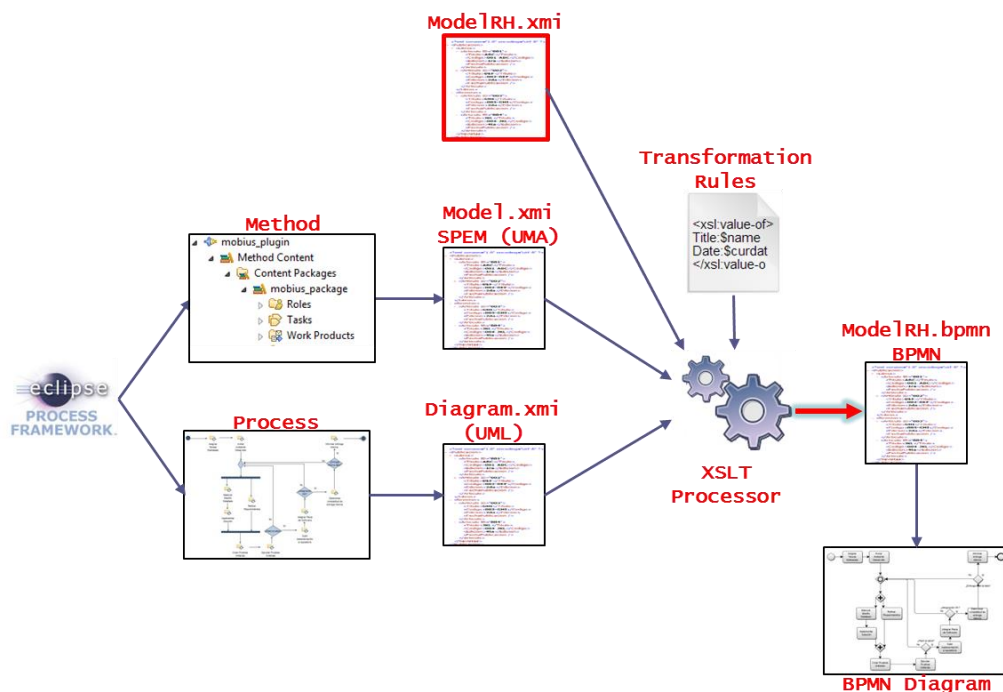
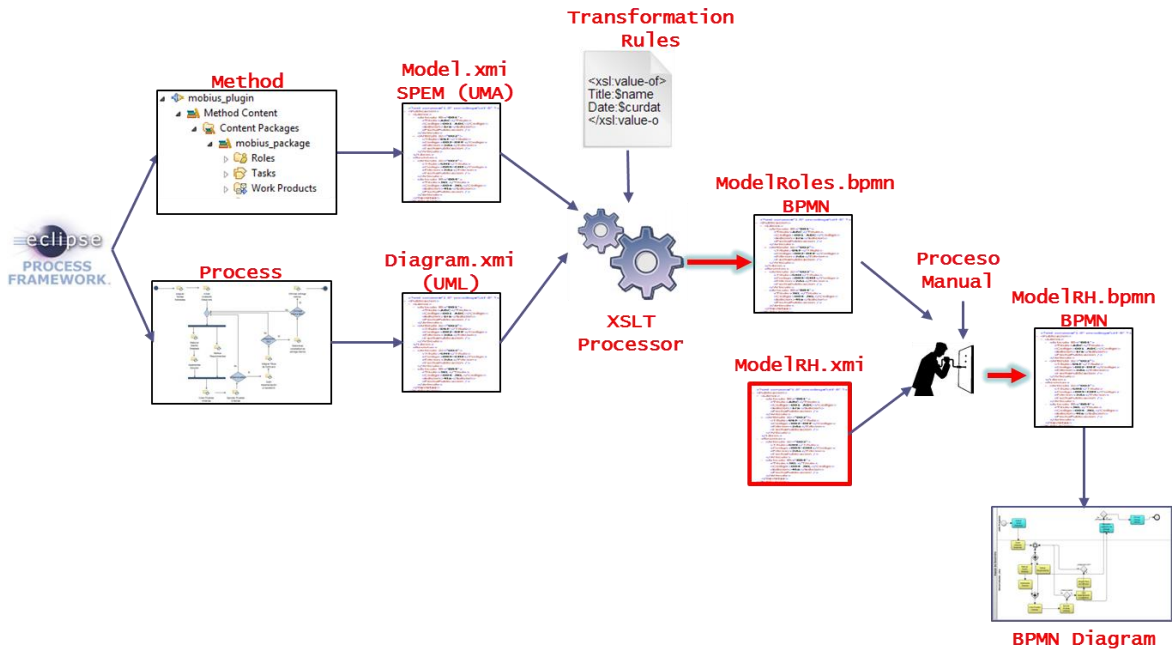


Figura 10b Esquema transformación SPEM-BPMN

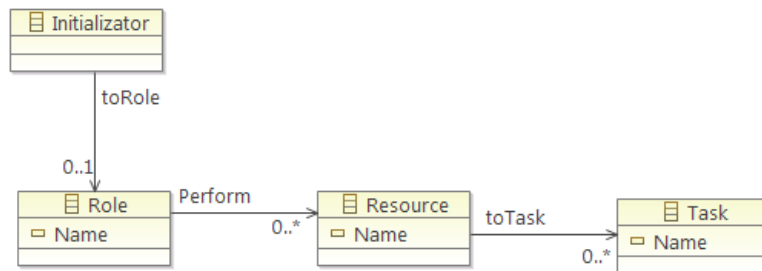


Para la implementación de la transformación se seleccionó el lenguaje XSLT, por las características descritas en la sección 2.5.

3.1 Metamodelo de Recursos Humanos

Con el propósito de representar las estructuras básicas Role, Resource y Task, se propone el metamodelo de recursos humanos, mostrado en la Figura 11.

Figura 11 Metamodelo de Recursos Humanos



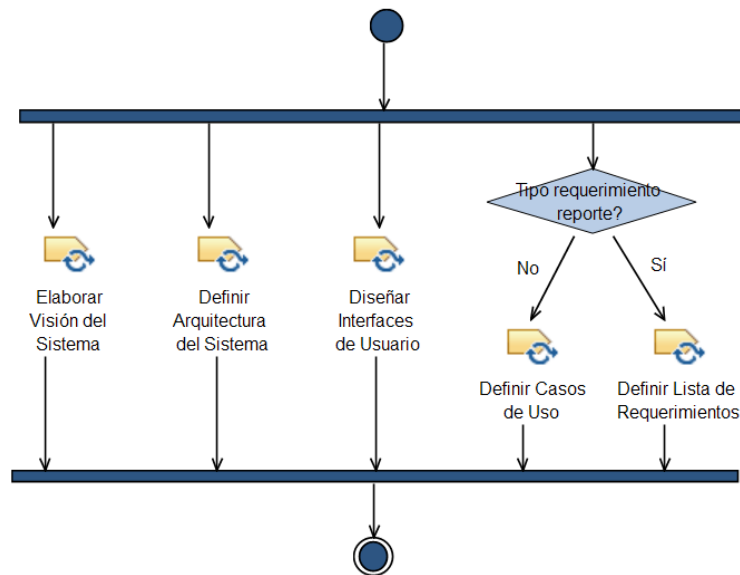
Conforme a este metamodelo se generan los modelos donde se instancian los recursos humanos que toman participación de un proyecto particular. Se vincula a cada recurso el rol desempeñado y las tareas asociadas.

3.2 Modelos Fuente -Procesos de Desarrollo en *SPEM*

La representación de cada una de las actividades de un proceso de desarrollo especificado en SPEM a través del editor EPF Composer, genera dos diferentes modelos: *model.xmi* y *diagram.xmi*, estos son los modelos fuente de la transformación.

Para presentar el propósito de cada uno de los modelos se ilustra el ejemplo de la actividad *Análisis de Requerimientos*, cuyo diagrama de actividad se muestra en la Figura 12. El diagrama de actividad muestra como *Análisis de Requerimientos* está compuesto por cinco tareas y un conjunto de nodos mediante el cual se expresa el flujo de control. Se incluyen nodos inicio, fin, join, fork y de decisión.

Figura 12 Diagrama de actividad *Análisis de Requerimientos*



Para analizar la estructura del modelo fuente *model.xmi* a continuación se presenta un fragmento del modelo asociado a la actividad *Análisis de Requerimientos*. En la línea 3 se muestra su atributo *name analisis_requerimientos*, a partir de la línea 5 se da inicio a la descripción de cada uno de los elementos *processElements*, iniciando con la especificación de un elemento tipo *TaskDescriptor*, cuyo *id* se establece en la línea 6, su atributo *name tsk_elab_vision* se fija en la línea 7, su *presentationName Elaborar Visión del Sistema* se establece en la línea 9, por último se establece en rol que ejecuta la tarea a través del atributo *performedPrimarilyBy* de la línea 14.

```

1 <org.eclipse.epf.uma:ProcessComponent
2   xmi:id="_8e4M4AylEeKgQ9IkPylC8w"
3   name="analisis_requerimientos"
4   guid="_8e4M4AylEeKgQ9IkPylC8w">
5     <processElements xsi:type="org.eclipse.epf.uma:TaskDescriptor"

```

```

6         xmi:id="_SGYg0Ay2EeKgQ9IkPylC8w"
7         name="tsk_elab_vision"
8         guid="_SGYg0Ay2EeKgQ9IkPylC8w"
9         presentationName="Elaborar Visión del Sistema"
10        superActivities="_8e4M4Qy1EeKgQ9IkPylC8w"
11        isSynchronizedWithSource="false"
12        mandatoryInput="_UJlutAy2EeKgQ9IkPylC8w _UJlutwy2EeKgQ9IkPylC8w"
13        output="_UJmVwAy2EeKgQ9IkPylC8w"
14        performedPrimarilyBy="_UJlusQy2EeKgQ9IkPylC8w">
15        <methodElementProperty xmi:id="_6sGb8IIjEeOizcckB-fz3wg"
16            name="me_references"
17            value="&lt;?xml version='1.0' encoding=';
18                standalone='no'?'&lt; &quot;/>" />
19        <Task href="uma:// 90rwU01HEEgke9eavCfiA# U0rvkAfTEeKis4GjYwzmcw"/>
20    </processElements>

```

Para describir el contenido del modelo *diagram.xmi* asociado a la actividad *Análisis de Requerimientos*, se incluye un fragmento de su especificación.

```

1 <node xmi:type="uml:InitialNode"
2     xmi:id="_wXGQ0Ay3EeKgQ9IkPylC8w"
3     outgoing="_UQwbwAy4EeKgQ9IkPylC8w"/>
4 <node xmi:type="uml:ForkNode"
5     xmi:id="_xGshwAy3EeKgQ9IkPylC8w"
6     outgoing="_VJmvUay4EeKgQ9IkPylC8w _V6xtkAy4EeKgQ9IkPylC8w
7         _YGbFwAy4EeKgQ9IkPylC8w _d8uKwAy4EeKgQ9IkPylC8w"
8     incoming="_UQwbwAy4EeKgQ9IkPylC8w"/>
9 <node xmi:type="uml:DecisionNode"
10    xmi:id="_hBKMIay4EeKgQ9IkPylC8w"
11    name="Tipo requerimiento reporte?"
12    outgoing="_qXY7sAy4EeKgQ9IkPylC8w _tQU3sAy4EeKgQ9IkPylC8w"
13    incoming="_YGbFwAy4EeKgQ9IkPylC8w"/>
14 <node xmi:type="uml:JoinNode"
15     xmi:id="_2lmlAAy4EeKgQ9IkPylC8w"
16     outgoing="_945McAy4EeKgQ9IkPylC8w"
17     incoming="_5nuSMay4EeKgQ9IkPylC8w _6MCgMAy4EeKgQ9IkPylC8w
18         _62J0MAy4EeKgQ9IkPylC8w _7lOgsAy4EeKgQ9IkPylC8w
19         _8XgSMay4EeKgQ9IkPylC8w"/>
20 <node xmi:type="uml:ActivityFinalNode"
21     xmi:id="_9MCtgAy4EeKgQ9IkPylC8w"
22     incoming="_945McAy4EeKgQ9IkPylC8w"/>
23 <edge xmi:type="uml:ControlFlow"
24     xmi:id="_UQwbwAy4EeKgQ9IkPylC8w"
25     source="_wXGQ0Ay3EeKgQ9IkPylC8w"
26     target="_xGshwAy3EeKgQ9IkPylC8w"/>
27 <edge xmi:type="uml:ControlFlow"
28     xmi:id="_VJmvUay4EeKgQ9IkPylC8w"
29     source="_xGshwAy3EeKgQ9IkPylC8w"
30     target="_XwVwCay3EeKgQ9IkPylC8w"/>

```

En las líneas 1,4,9, 14 y 20 se especifica cada uno de los nodos que determinan el flujo de control de la actividad. A partir de la línea 23 se especifican las entidades *edge* que son las líneas conectoras entre cada una de las entidades del diagrama, bien sean tareas o nodos de control.

3.3 Diseño de la Transformación de Modelos SPEM a BPMN

Las correspondencias entre las entidades de los dos metamodelos involucrados en la transformación se presentan en la Tabla 8, (adaptado de [43]).

SPEM, al ser un metamodelo orientado a soportar específicamente procesos de software es capaz de representar las estructuras genéricas de procesos (*Process, LifecycleModel, Combination, Activity, Artifact, Resource, Procedure, PatternofActivity, Restrictions*). Por el contrario, BPMN al tener una orientación más amplia, dirigida a la representación de procesos de negocio, permite una menor especificidad para representar procesos de desarrollo de software. Por lo anterior, estructuras de SPEM como *process, processComponent phase* y *step* no encuentran un elemento correspondiente en el metamodelo BPMN.








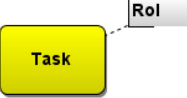








Tabla 8 Mapeo entidades entre SPEM y BPMN





Estructura Estándar de Proceso	SPEM	BPMN
Process	Process ProcessComponent	
LifecycleModel	Process Iteration	Embedded Sub-Process
Combination	Phase	
Activity	Discipline Activity	Embedded Sub-Process
	TaskUse	Task
	Step	
Artifact	WorkProductDefinition WorkProductUse	Data Object
Resource	RoleUse RoleDefinition	Pool Lane
Procedure	Guidance Guideline ToolMentor Template Checklist	Text Annotation
PatternofActivity	Step	

Restrictions	WorkSequence ContentDescription Goal Precondition WorkDefinitionParameter Category WorkProductRelationship	Rule
--------------	--	------

Las reglas de transformación definidas en la implementación aplican sobre todos los elementos incluidos en la Tabla 9. La transformación tiene como entrada los elementos de la columna SPEM y como salida los elementos de la columna BPMN.

Tabla 9 Relaciones implementadas en la Transformación

Elemento del Proceso	SPEM	BPMN
Start Node		
End Node		
Task		
Rol		
Activity		
PatternCapability		
Decision node		
Merge Node		

Fork Node		
Join Node		

Los roles que participan en el proceso definido en SPEM son representados en el diagrama BPMN a través de notas vinculadas a cada tarea. Otro escenario posible era la representación de los roles a través de *pools* y *lanes*, distribuyendo las tareas de acuerdo con los respectivos roles. Ambas representaciones son posibles, pero no se pueden obtener de forma simultánea. La representación de roles a través de notas permite además la transformación de entidades *Activity* de SPEM a entidades *Subprocess* de BPMN, este último fue el criterio que justificó la elección.

3.4 Implementación de la Transformación de Modelos SPEM a BPMN

El uso de XSLT supone un conjunto de restricciones. Inicialmente, el procesador XSLT aplica reglas de transformación sobre un solo modelo fuente. Los procesos de desarrollo en EPF Composer se definen a partir de dos modelos: el primero con la información referente al contenido de método o *content method* (*model.xml*), y el segundo con la representación gráfica a través de los estereotipos definidos por el perfil UML de SPEM *activity diagram* (*diagram.xml*).

En el *model.xml* persiste la información de todas las entidades como nodos con estructura. El *diagram.xml* contiene solamente conectores sin estructura.

Al iniciar la transformación los dos modelos fuente se componen en un solo modelo para superar la restricción del procesador XSLT. La transformación comienza recorriendo la estructura de nodos (actividades del proceso) con el propósito de encontrar el nodo raíz (actividad raíz o padre).

El recorrido a través de los conectores (*diagram.xml*) y la contrastación de la estructura (en el *model.xml*) permiten recuperar cada una de las entidades que describen el comportamiento del proceso de desarrollo, tales como fases, actividades, tareas, eventos (inicio/fin) y nodos de decisión simples y compuestos (*join*, *merge* y *fork*). Conforme se recuperan las entidades del modelo de entrada se aplican las reglas de transformación para el mapeo a entidades en BPMN que permitan la construcción del modelo de salida. La conexión de los objetos en

BPMN y el establecimiento del flujo del proceso de negocio resultante son posibles recuperando la información del *diagram.xmi*.

Para conducir de manera explícita el proceso de selección sobre el conjunto de nodos, se hace uso de la primitiva `<xsl:for-each>`. Ésta genera explícitamente un bucle para una colección de nodos.

La sintáxis es la siguiente:

```
<xsl:for-each select=" " >
    .....
</xsl:for-each>
```

El atributo `select` puede contener cualquier ruta XPath y el bucle reunirá a cada elemento del conjunto de nodos resultante.

Otro recurso que se utilizó para el procesamiento condicional es la sentencia `<xsl:if>`. Esta permite evaluar expresiones lógicas (según defina la especificación XPath) y a partir de su resultado producir diferentes salidas.

Sintaxis:

```
<xsl:if test="expresión booleana">
    <!--Plantilla -- >
</xsl:if>
```

A continuación, en la el fragmento del código XSLT con el que se procesan los nodos *fork* y *join* de SPEM, para ser transformados a nodos *Gateway* de tipo *Diverging* y *converging* respectivamente.

```
<xsl:when test="(@xmi:type = 'uml:ForkNode') or (@xmi:type = 'uml:JoinNode')">
  <parallelGateway xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
    <xsl:choose>
      <xsl:when test="@xmi:type = 'uml:ForkNode'">
        <xsl:attribute name="gatewayDirection">Diverging</xsl:attribute>
      </xsl:when>
      <xsl:when test="@xmi:type = 'uml:JoinNode'">
        <xsl:attribute name="gatewayDirection">Converging</xsl:attribute>
      </xsl:when>
    </xsl:choose>
    <xsl:attribute name="name"><xsl:value-of select="@name" /></xsl:attribute>
    <xsl:attribute name="id"><xsl:value-of select="@xmi:id" /></xsl:attribute>
    <xsl:if test="string-length(@incoming) != 0">
      <xsl:call-template name="incomingtokenize">
        <xsl:with-param name="text" select="@incoming"/>
        <xsl:with-param name="separator" select="' '"/>
      </xsl:call-template>
    </xsl:if>
    <xsl:if test="string-length(@outgoing) != 0">
      <xsl:call-template name="outgoingtokenize">
        <xsl:with-param name="text" select="@outgoing"/>
        <xsl:with-param name="separator" select="' '"/>
      </xsl:call-template>
    </xsl:if>
```



```

    </parallelGateway>
  </xsl:when>

```

Por último, se presenta el fragmento de código XSLT que permite recorrer todas las tareas (*taskDescriptor*) de SPEM y recuperar el rol (*performedPrimarilyBy*) responsable de su ejecución. Posteriormente se genera una nota adjunta (*textAnnotation*) con el nombre del rol y ésta se asocia a la tarea en el diagrama BPMN.

```

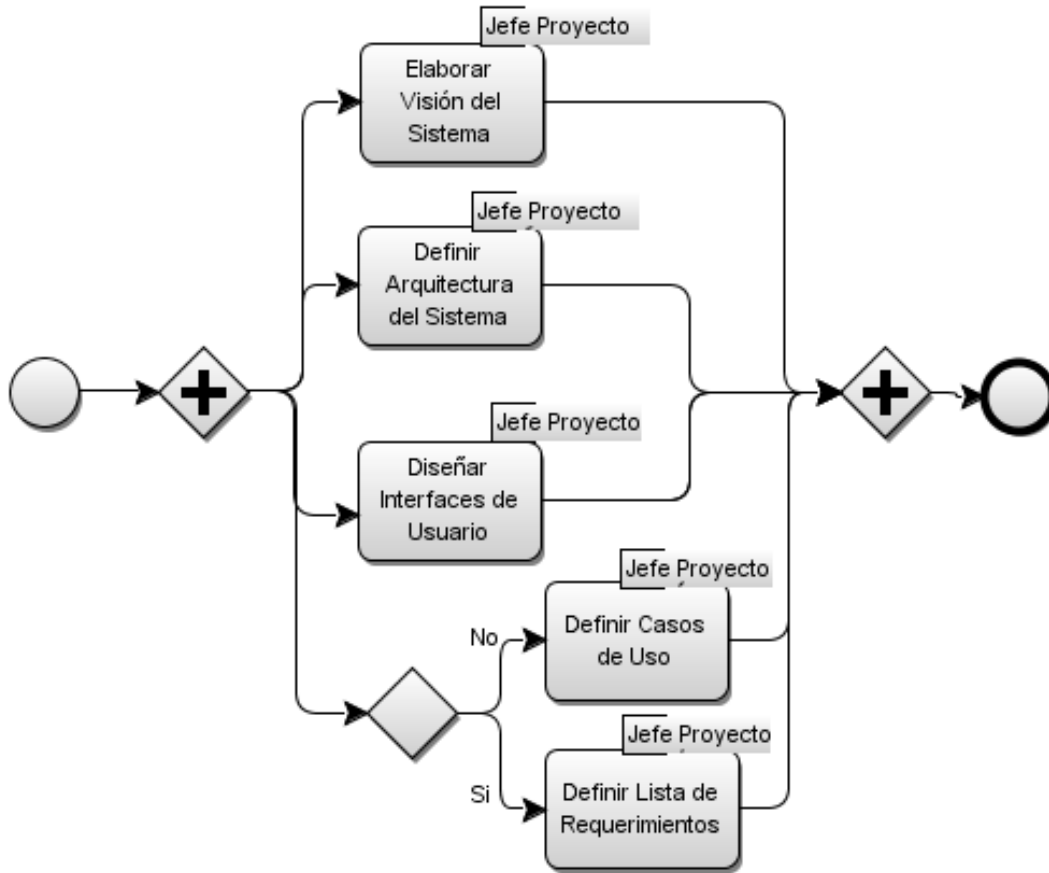
<xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/processElements">
  <xsl:if test="(@xsi:type = 'org.eclipse.epf.uma:TaskDescriptor') and (@presentationName =
    $name)">
    <xsl:variable name="idRol" select="@performedPrimarilyBy"/>
    <xsl:for-each select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent
      /processElements">
      <xsl:if test="(@xsi:type='org.eclipse.epf.uma:RoleDescriptor') and (@xmi:id =
        $idRol)">
        <textAnnotation textFormat="text/plain"
          xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
          <xsl:attribute name="id"><xsl:value-of select="concat('N_', $id)"
            /></xsl:attribute>
          <text xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
            select="@presentationName" /></text>
        </textAnnotation>
        <association xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
          associationDirection="None">
          <xsl:attribute name="id"><xsl:value-of select="concat('A_', $id)"
            /></xsl:attribute>
          <xsl:attribute name="sourceRef"><xsl:value-of select="concat('N_', $id)"
            /></xsl:attribute>
          <xsl:attribute name="targetRef"><xsl:value-of select="$id"
            /></xsl:attribute>
        </association>
      </xsl:if>
    </xsl:for-each>
  </xsl:if>
</xsl:for-each>

```

3.5 Modelos Objetivo - Procesos de Negocio en BPMN

Los modelos objetivo en la transformación implementada en el presente trabajo son conformes con BPMN. En la **¡Error! No se encuentra el origen de la referencia.** se presenta el diagrama en BPMN correspondiente al Proceso de Negocio *Análisis de Requerimientos*. Los modelos fuente conformes a SPEM fueron estudiados en la sección 3.3 y su diagrama de actividad se muestra en la Figura 14.

Figura 13 Diagrama del Proceso de Negocio *Análisis de Requerimientos*



Para ilustrar la estructura de los modelos BPMN se presenta un fragmento del modelo BPMN producto de transformar la actividad *Análisis de Requerimientos*.

En la línea 2 se define un elemento tipo *task* cuyo nombre se fija en la línea 7, bajo el atributo *name* *Elaborar Visión del Sistema*. Entre las líneas 8 y 11 se especifican los conectores que entran y salen a la tarea. La especificación de la tarea *Definir Arquitectura del Sistema* se presenta entre las líneas 13 y 23. Especificadas todas las tareas del proceso se inicia con la especificación de los nodos, el primero es nodo *fork* que aparece en el diagrama BPMN luego del nodo inicio. Este elemento de tipo *parallelGateway* se especifica entre las líneas 24 y 37, su definición incluye el parámetro *gatewayDirection* con valor *Diverging* para indicar que es de tipo *fork*, además se establecen todos los conectores que salen de este. Entre las líneas 38 y 42 se especifica el nodo de inicio *startEvent*. Una vez especificados todos los nodos se especifican los conectores.

```

1 <process isClosed="false" isExecutable="true" processType="None">
2   <task xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
3     completionQuantity="1"
4     isForCompensation="false"
5     startQuantity="1"
6     id="_-MUwZIExEeOZWP842uFiMA"
7     name="Elaborar Visión del Sistema">
8     <incoming xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
9       MUwf4ExEeOZWP842uFiMA</incoming>
10    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
11      MUwh4ExEeOZWP842uFiMA</outgoing>
12  </task>
13  <task xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
14    completionQuantity="1"
15    isForCompensation="false"
16    startQuantity="1"
17    id="_-MUwaIExEeOZWP842uFiMA"
18    name="Definir Arquitectura del Sistema">
19    <incoming xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
20      MUwgIExEeOZWP842uFiMA</incoming>
21    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
22      MUwiIExEeOZWP842uFiMA</outgoing>
23  </task>
24  <parallelGateway xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
25    gatewayDirection="Diverging"
26    name="" id="_-MUweIExEeOZWP842uFiMA">
27    <incoming xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
28      MUwfYExEeOZWP842uFiMA</incoming>
29    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
30      MUwf4ExEeOZWP842uFiMA</outgoing>
31    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
32      MUwgIExEeOZWP842uFiMA</outgoing>
33    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
34      MUwgYExEeOZWP842uFiMA</outgoing>
35    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
36      MUwgoExEeOZWP842uFiMA</outgoing>
37  </parallelGateway>
38  <startEvent xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
39    id="_-MUweoExEeOZWP842uFiMA">
40    <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">_-
41      MUwfYExEeOZWP842uFiMA</outgoing>
42  </startEvent>
43  <sequenceFlow xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
44    id="_-MUwfYExEeOZWP842uFiMA" name="" sourceRef="_-
45    MUweoExEeOZWP842uFiMA"
46    targetRef="_-MUweIExEeOZWP842uFiMA" />

```

Capítulo 4. Validación de la Transformación

En este capítulo se describe el proceso de validación aplicado a la transformación desarrollada como parte de este proyecto.

Inicialmente, y teniendo como modelo fuente el proceso de desarrollo de software de una empresa real (en SPEM), se realiza la transformación de cada una de las actividades a procesos de negocio en BPMN. Se replica el ejercicio de transformación sobre las mismas actividades del proceso de desarrollo, pero esta vez se utiliza como herramienta de transformación MOSKitt4ME, que permite transformaciones en el mismo sentido (SPEM - BPMN). Una primera validación consistió en comparar los modelos producidos por las diferentes herramientas. En el siguiente ejercicio, los procesos de negocio generados por la transformación implementada fueron inspeccionados uno a uno por el autor de este trabajo, para validar su consistencia con las actividades originales del proceso de desarrollo. Los resultados se presentan en las secciones siguientes.

Por último, y en procura de eliminar algún sesgo inducido al ser el autor el mismo que realiza la validación, se consultó a un experto vinculado a la gerencia de desarrollo de la empresa. A este se le compartieron los modelos BPMN de cada proceso de negocio, y tras una inspección encontró que guardaban total coherencia con los definidos en el proceso de desarrollo de la empresa.

Un interesante escenario para una validación, no solamente de la herramienta de transformación, sino de la idea subyacente del proyecto, es la ejecución de los procesos de negocio obtenidos sobre una plataforma BPMS (como Bizagi⁶ o Bonita⁷). Para lograr esta ejecución, si bien el proceso de negocio es indispensable, también es necesario definir un conjunto de parámetros relativos a la automatización y el control del proceso. Los parámetros necesarios son entre otros: reglas dinámicas de asignación, datos que se van a manejar, reglas de negocio para la automatización de decisiones, integración para el intercambio de información con

⁶ <http://www.bizagi.com/>

⁷ <http://www.bonitasoft.com/>

otras aplicaciones, alarmas e indicadores. Dado que una definición precisa del conjunto de parámetros requeridos escapa del alcance de este proyecto, y hacer suposiciones sobre el mismo pudiera influir significativamente en el comportamiento de los resultados del experimento, resolvimos incluir esta validación dentro del trabajo futuro.

4.1 Caso de Estudio

La transformación se aplicó sobre el proceso de desarrollo de la compañía Mobuis. Mobuis es una joven empresa Chilena que tiene como principal nicho de negocio el desarrollo de soluciones integradas de software y hardware para la administración y el control de grandes empresas del transporte público de Santiago.

Mobius tiene aproximadamente tres años de antigüedad. Cuenta con más de 30 empleados, fundamentalmente ingenieros en computación y técnicos en electrónica. Desde hace un año han comenzado con la formalización de su proceso de desarrollo, pero una vez alcanzada esta meta, desean automatizar la ejecución de su proceso a fin de agilizar esta tarea y permitir un mayor control sobre la gestión.

El proceso de desarrollo que guía los proyectos al interior de la compañía reúne conceptos de RUP y los mezcla con algunas prácticas de metodologías ágiles. Esta combinación le permite alcanzar un proceso formal pero con el suficiente dinamismo que le permita responder a proyectos de corta y media duración.

En el proceso de desarrollo se definieron cuatro fases, (inicio, elaboración, construcción y transición), cinco disciplinas (administración, arquitectura, desarrollo, requerimientos y testing) y once actividades. Visto a mayor detalle, el proceso incluye 47 tareas, 10 roles y 44 productos de trabajo, sin embargo se identificaron puntos de variabilidad que posibilitan el ajuste del proceso al contexto de cada proyecto.

4.2 Restricciones del experimento

La transformación implementada se valida sobre un único proceso de software. Este proceso al estar formalizado e implementado en una empresa real reúne las condiciones de complejidad suficientes en cuanto a su dimensión, comportamiento y estructura para considerarlo un escenario de prueba adecuado.

La decisión de transformar entidades *Activity* de SPEM a entidades *Subprocess* de BPMN y desplegar estos *Subprocess* de modo extendido imposibilita el uso de *pools* y *lanes*, por lo cual se representan los roles a través de notas adjuntas a cada tarea.

Los diagramas BPMN producidos con la transformación implementada son comparados en todos los casos con los diagramas producidos por la herramienta MOSKitt4ME. Si bien en el estado del arte se encontraron otros trabajos que referían diferentes transformaciones; éstas, en su mayoría, son solamente formalizaciones y no están disponibles como productos que permitan su uso.

4.3 Transformación del Proceso de Desarrollo de Software de la empresa Mobius

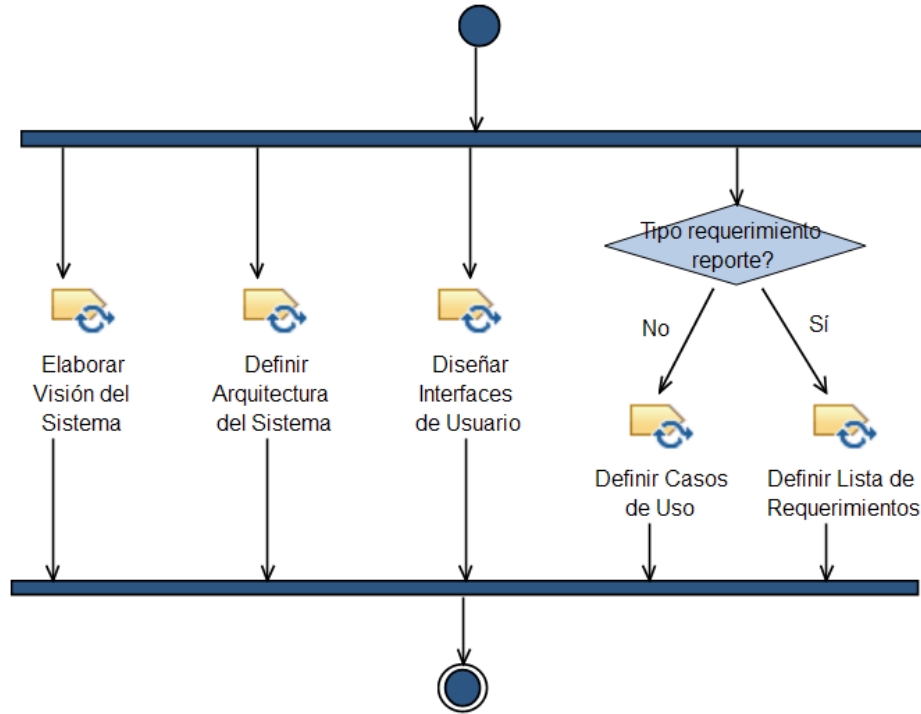
A lo largo de esta sección se presentan una a una las actividades principales que conforman el proceso de desarrollo de la empresa Mobius. Para cada actividad se muestra el diagrama UML de actividad generado en EPF Composer, el diagrama BPMN obtenido a través de la transformación MOSKitt4ME y por último, el diagrama BPMN obtenido aplicando la transformación implementada.

4.3.1 Transformación de la actividad *Análisis de Requerimientos*

El propósito de esta actividad es el descubrimiento, modelado y especificación gruesa de los requerimientos del sistema; a la vez se elaboran las primeras versiones de los documentos de visión y arquitectura.

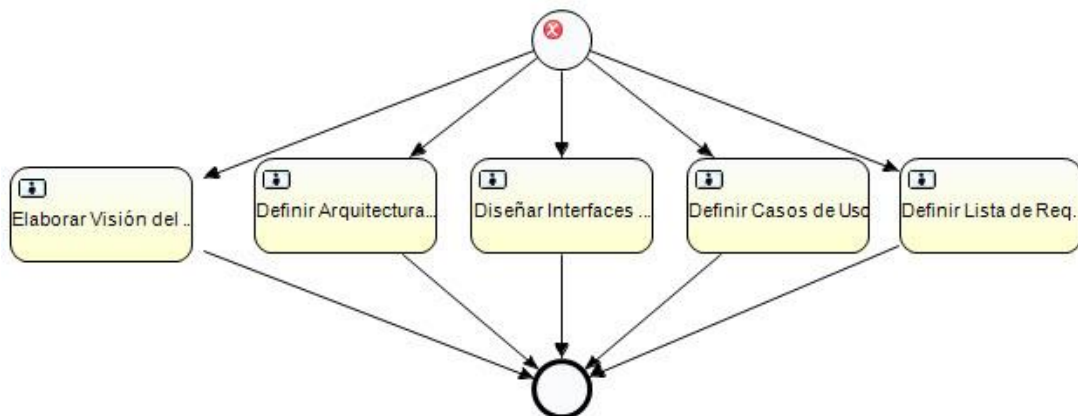
La Figura 14 presenta el diagrama de actividad. En el diagrama se hace uso de nodos *fork* y *join* para indicar que las tareas se inician en paralelo y deben terminar concurrentemente para dar por finalizada la actividad. Así mismo, se hace uso de un nodo de decisión, que indica que la evaluación de una condición determinará el flujo a seguir.

Figura 14 Diagrama de la actividad *análisis de requerimientos* en EPF Composer



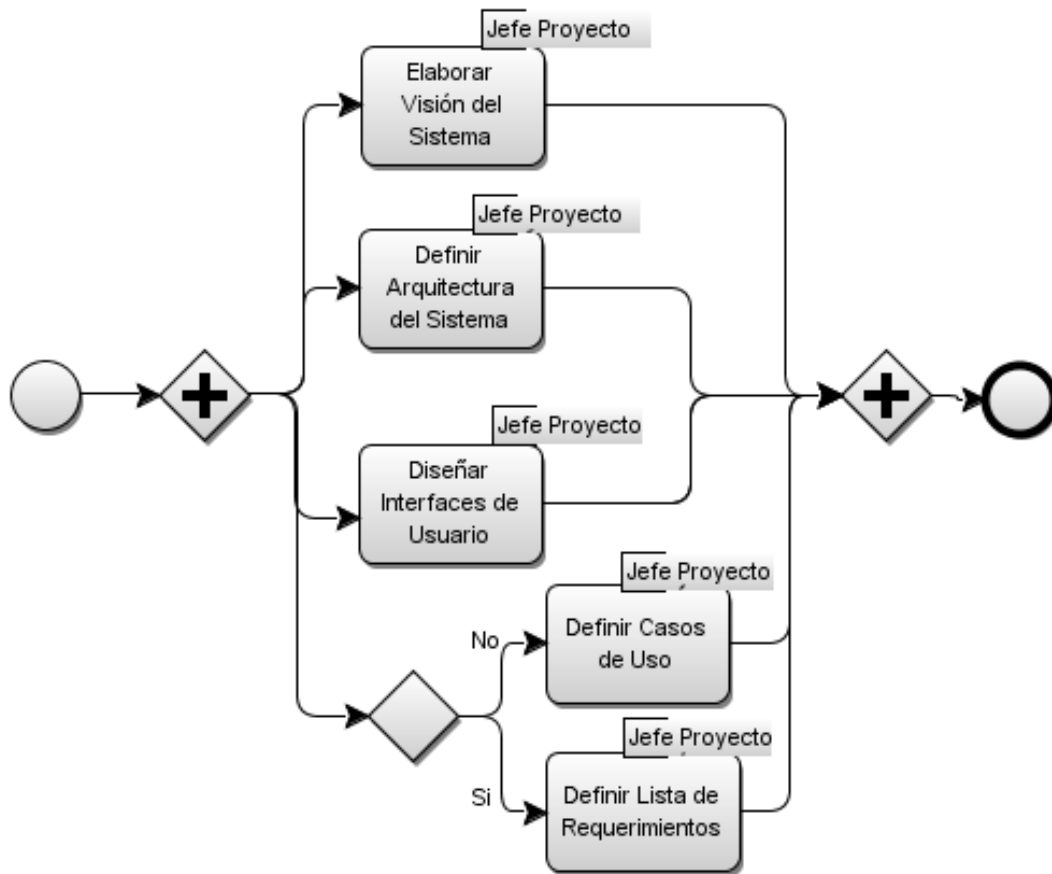
En la Figura 15 se presenta el diagrama BPMN producido por la transformación de Moskkit4me. Es importante destacar que el diagrama guarda coherencia en la cantidad de las tareas y en el nombre de cada una de ellas. El diagrama, sin embargo, no representa las condiciones de concurrencia para el inicio y fin de las tareas, ni tampoco incluye el nodo de decisión con lo cual se cambia el flujo del proceso.

Figura 15 Diagrama BPMN de la actividad *análisis de requerimientos* obtenida con MOSKitt4ME



En la Figura 16 se presenta el diagrama BPMN obtenido a través de la transformación implementada. El diagrama además de ser consistente en la cantidad de tareas y en el nombre de cada una de ellas, relaciona, a través de una nota adjunta, el rol responsable de su ejecución. Representa también las condiciones de concurrencia para el inicio y fin de las tareas a través de las compuertas. Por último, es de destacar que se incluye el nodo de decisión junto a las etiquetas (Si/No) con lo cual se mantiene el flujo del proceso original.

Figura 16 Diagrama BPMN de la actividad *análisis de requerimientos* obtenida con la transformación implementada



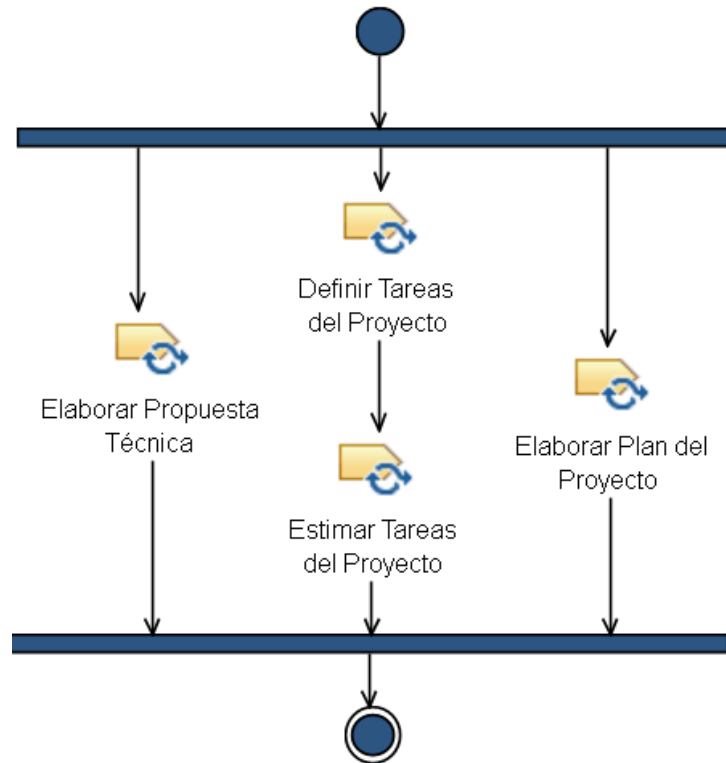
4.3.2 Transformación de la actividad *Planificación del Proyecto*

El objetivo de la actividad *planificación del proyecto* de software es proporcionar un marco de trabajo que permita realizar estimaciones razonables de recursos, costos y planificación temporal.

El diagrama de actividad mostrado en la Figura 17 incluye la ejecución de cuatro tareas que se inician concurrentemente. Una condición particular del flujo es que la tarea *Definir Tareas*

Proyecto antecede a *Estimar Tareas del Proyecto*; así, el fin de la primera tarea dará inicio a la segunda. Por último, el nodo *join* indica que deben haberse ejecutado todas las tareas concurrentes para terminar la actividad.

Figura 17 Diagrama de la actividad *planificación de proyecto* en EPF Composer



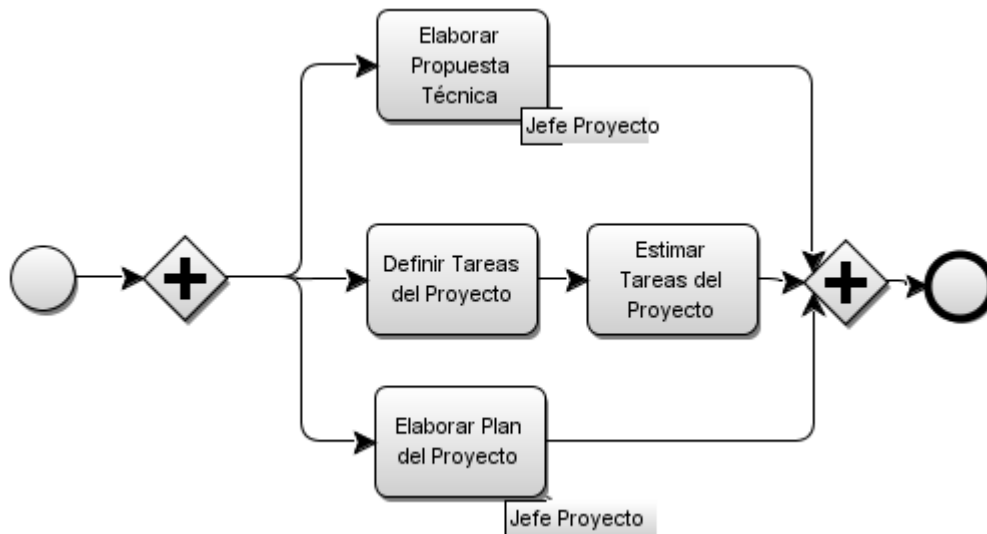
El diagrama logrado con Moskkitt4me (Figura 18) representa acertadamente la relación de antecesor/sucesor entre las tareas *Definir Tareas Proyecto* y *Estimar Tareas del Proyecto*. Sin embargo, la condición de concurrencia para el inicio y el fin de las tareas no se logra; ésta será la constante cada vez que en el modelo SPEM se vinculen nodos *fork* y *join* ya que MOSKitt4ME no logra su transformación.

Figura 18 Diagrama de la actividad *planificación de proyecto* obtenida con MOSKitt4ME



En el diagrama BPMN de la Figura 19 se hace uso de compuertas para representar correctamente los nodos *fork* y *join* del proceso fuente de la transformación. Del mismo modo, se mantiene el comportamiento de flujo entre las tareas *Definir Tareas del Proyecto* y *Estimar Tareas del Proyecto*.

Figura 19 Diagrama BPMN de la actividad *planificación de proyecto* obtenida con la transformación implementada

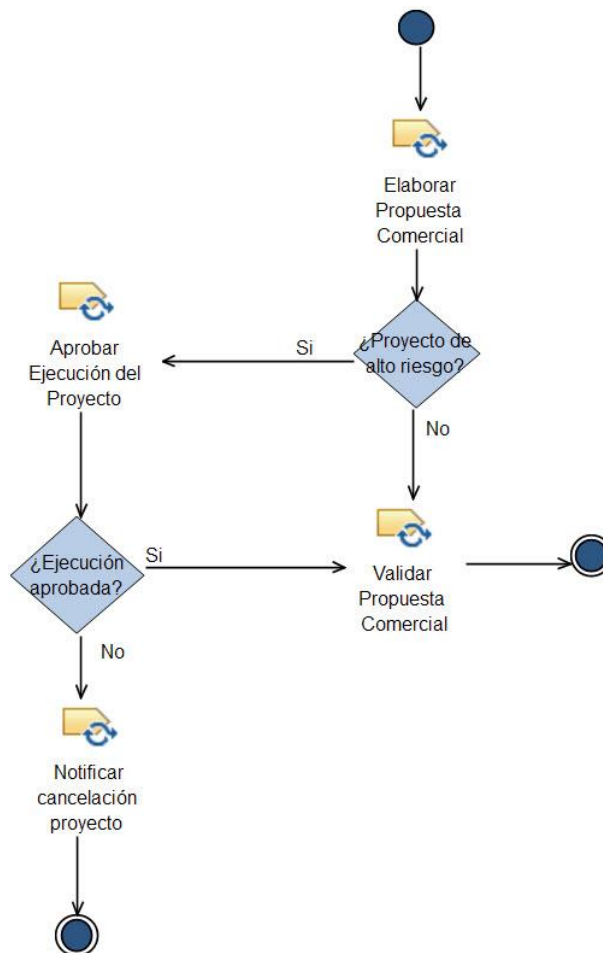


4.3.3 Transformación de la actividad *Acuerdo Comercial*

Esta actividad involucra las tareas relativas al convenio entre cliente y empresa, para alcanzar una posición común sobre las condiciones de ejecución, alcance, tiempo y costo del proyecto.

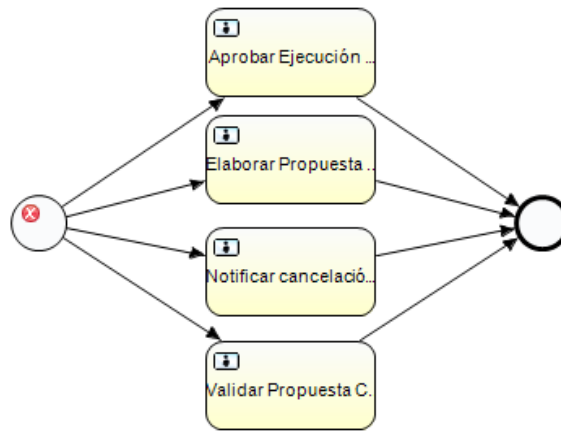
La actividad *Acuerdo Comercial* incluye cuatro tareas y su flujo está definido a partir de dos nodos de decisión como se muestra en la Figura 20

Figura 20 Diagrama de la actividad *acuerdo comercial* en EPF Composer



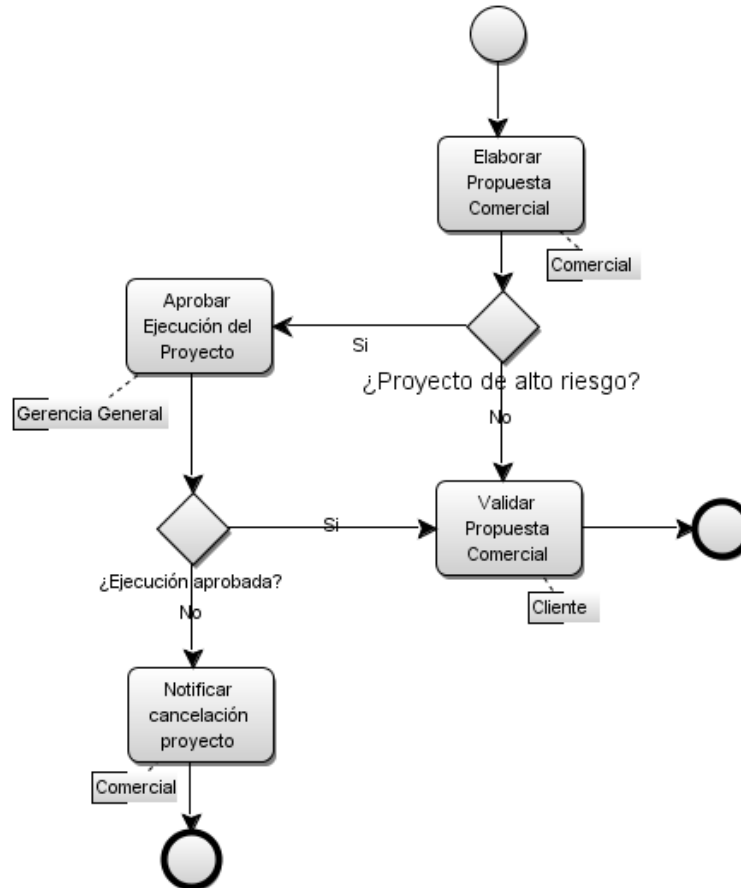
El diagrama BPMN producido por Moskkit4me incluye de manera correcta las cuatro tareas, pero no representa el flujo de la actividad fuente, como puede apreciarse en la Figura 21

Figura 21 Diagrama BPMN de la actividad *acuerdo comercial* obtenida con MOSKitt4ME



El diagrama producto de la transformación implementada representa de manera adecuada las tareas de la actividad *acuerdo comercial* y las relaciones entre estas, como se muestra en la figura Figura 22

Figura 22 Diagrama BPMN de la actividad *acuerdo comercial* obtenida con la transformación implementada

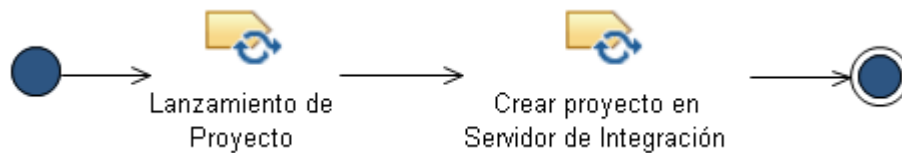


4.3.4 Transformación de la actividad *Lanzamiento*

En esta actividad se genera una ficha del proyecto, se realizan modificaciones al plan de iteraciones y al plan de proyecto. Con esta actividad se cierra la fase de *Inicio del proceso*.

Es una de las actividades más sencillas del proceso ya que incluye solamente la ejecución en secuencia de dos tareas, como se ilustra en la Figura 23 .

Figura 23 Diagrama de la actividad *lanzamiento* en EPF Composer



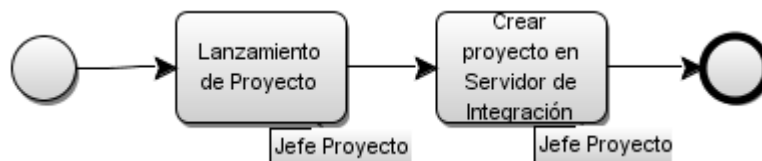
Se consigue un diagrama correcto con el uso de MOSKitt4ME, como se muestra en la Figura 24

Figura 24 Diagrama BPMN de la actividad *lanzamiento* obtenida con MOSKitt4ME



También se consigue un diagrama correcto con el uso de la transformación implementada, como se muestra en la Figura 25

Figura 25 Diagrama BPMN de la actividad *lanzamiento* obtenida con la transformación implementada



4.3.5 Transformación de la actividad *Validación de Arquitectura*

Esta actividad se realiza en la fase de *Elaboración* del proceso de desarrollo de la empresa. Su objetivo es la definición, implementación y validación de un prototipo de arquitectura del sistema.

La actividad mostrada en la Figura 26 está conformada por cuatro tareas. La primera tarea en ejecutarse es *Definir Prototipo de Arquitectura*, concluida ésta, se da inicio concurrente a las

tareas *Implementar Prototipo de Arquitectura* y *Diseñar casos de prueba de prototipo de arquitectura* estas dos tareas deben finalizar para dar inicio a *Validar Prototipo de Arquitectura*. El diagrama generado por MOSKitt4ME (Figura 27) no expresa las condiciones de concurrencia, lo que sí se logra en el diagrama generado por la transformación implementada, como se evidencia en la Figura 28.

Figura 26 Diagrama de la actividad *validación de arquitectura* en EPF Composer

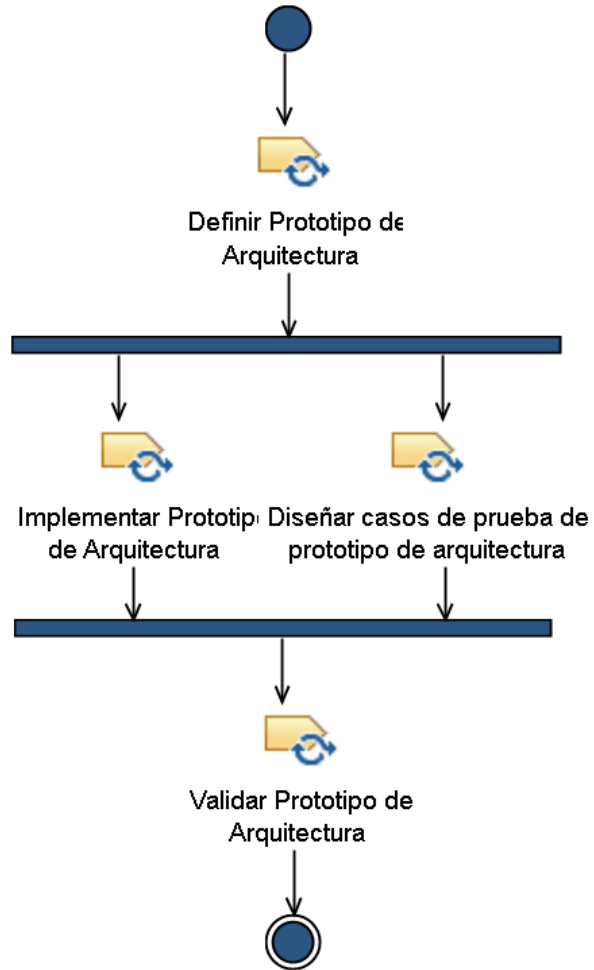


Figura 27 Diagrama BPMN de la actividad *validación de arquitectura* obtenida con MOSKitt4ME

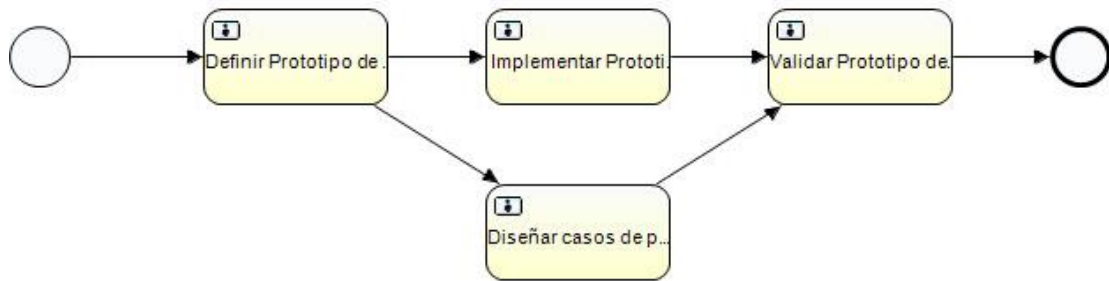
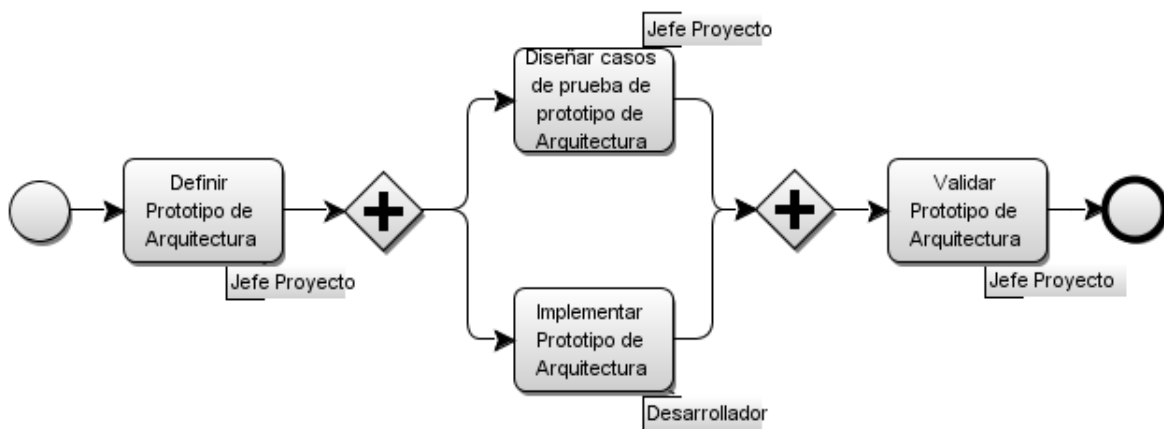


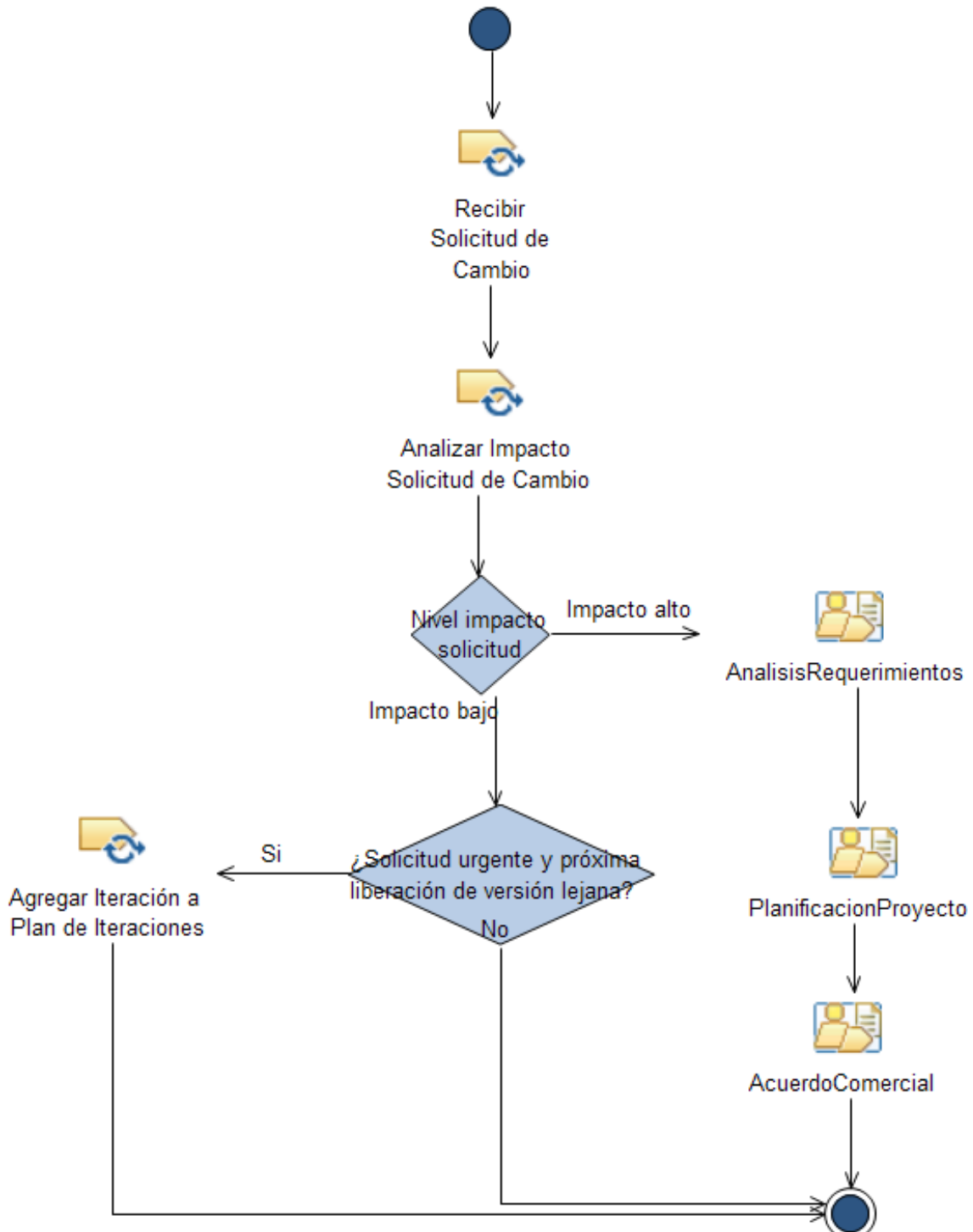
Figura 28 Diagrama BPMN de la actividad *validación de arquitectura* obtenida con la transformación implementada



4.3.6 Transformación de la actividad *Control de Cambios*

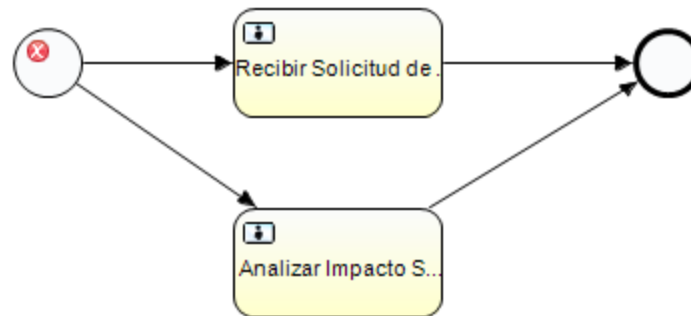
Esta actividad se realiza en la fase de construcción. Comprende todas las tareas desde el ingreso de una nueva solicitud de cambio hasta la generación de una propuesta comercial. Control de cambios es una actividad compuesta, que además de incluir tareas, incluye también a otras actividades como son: *Análisis de Requerimiento*, *Planificación de Proyecto* y *Acuerdo Comercial*. La Figura 29 ilustra el contenido de la actividad.

Figura 29 Diagrama de la *control de cambios* en EPF Composer



Al intentar la transformación con MOSKitt4ME se obtiene el diagrama BPMN mostrado en la Figura 30. El diagrama resultante incluye solamente dos tareas, lo que claramente no representa ni en contenido ni el comportamiento del proceso fuente en SPEM.

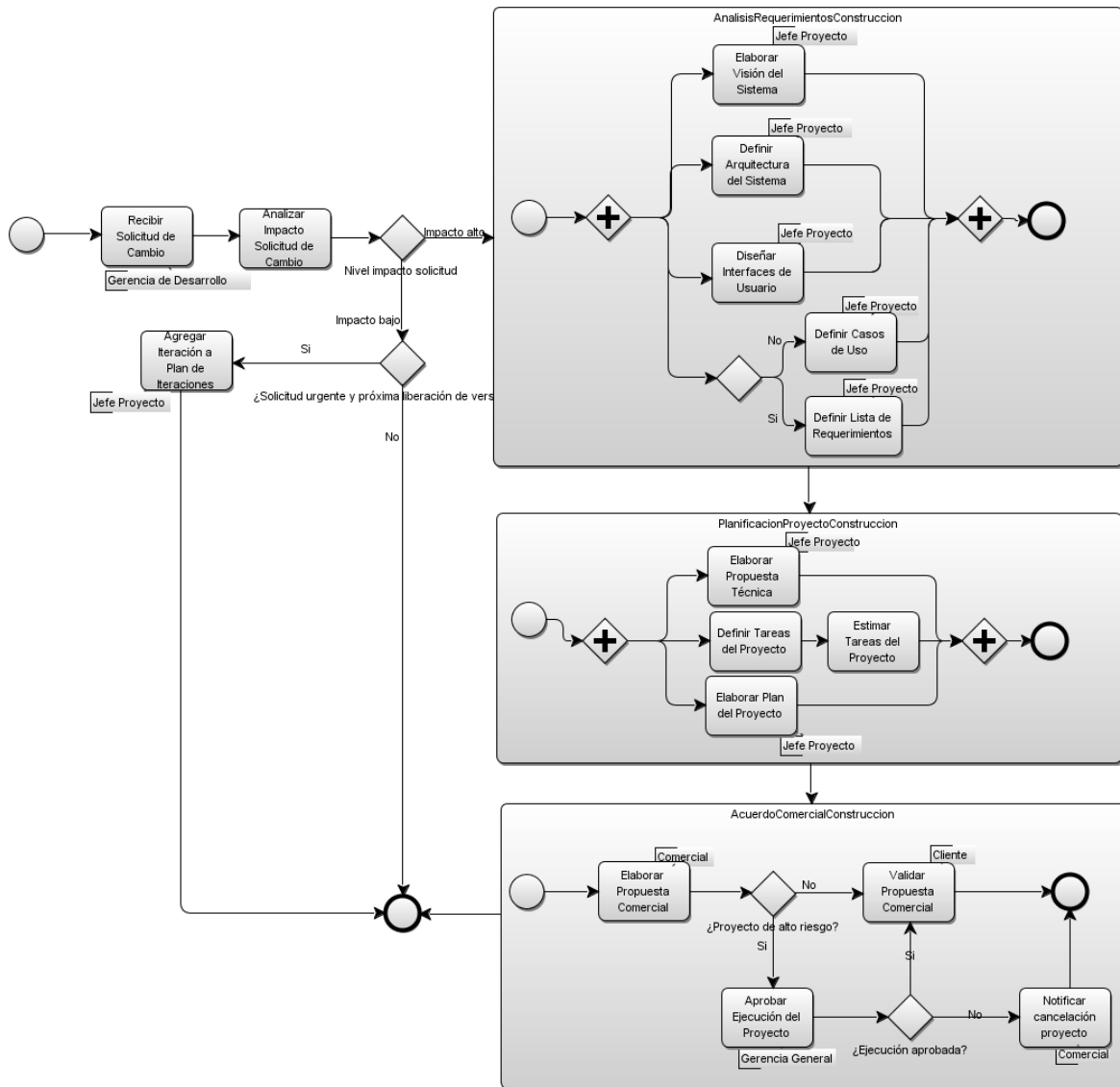
Figura 30 Diagrama BPMN de la actividad *control de cambios* obtenida con MOSKitt4ME



En la implementación de la transformación se definieron reglas para la conversión de entidades *Activity* de SPEM a entidades *Subprocess* de BPMN. La visualización del *Subprocess* dentro del diagrama puede realizarse en dos modos distintos: de un lado, un despliegue extendido donde se incluyen todos los elementos del *Subprocess* en el diagrama principal, o bien un modo reducido; donde se incluye el *Subprocess* pero sin su contenido, dejándolo solamente indicado en el diagrama; en este trabajo se optó por la primera opción.

La Figura 31 muestra el diagrama BPMN obtenido a través de la transformación implementada. Se incluyen los *Subprocess* *Análisis de Requerimiento*, *Planificación de Proyecto* y *Acuerdo Comercial*. Para cada *Subprocess* se incluyen los nodos inicio y fin, las tareas, roles, compuertas y conectores para representar enteramente las entidades equivalentes del modelo original SPEM.

Figura 31 Diagrama BPMN de la actividad *control de cambios* obtenida con la transformación implementada



4.3.7 Transformación de la actividad *Diseño de Pruebas*

Diseño de pruebas reúne las tareas relativas a la especificación y validación de casos de prueba. En la Figura 32 se ilustra su diagrama de actividad que incluye un nodo de decisión; de acuerdo al cumplimiento de la condición el flujo retorna a la ejecución de la tarea *Diseñar Casos de Prueba*.

El diagrama generado por MOSKitt4ME (Figura 33) no representa el proceso original dado que no incluye el nodo de decisión, mientras que el diagrama producido por la transformación implementada (Figura 34) logra hacerlo de forma adecuada.

Figura 32 Diagrama de *diseño de pruebas* en EPF Composer

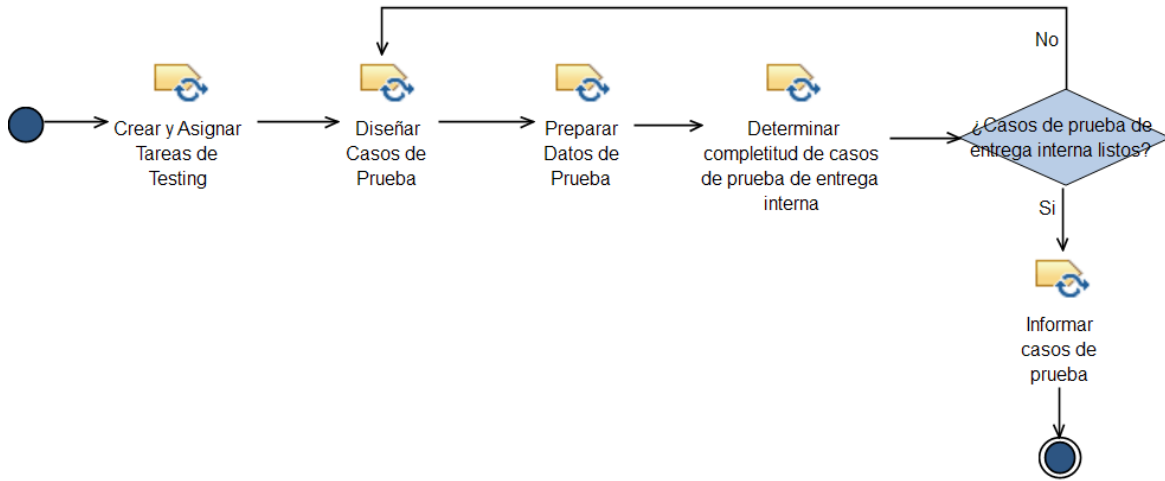


Figura 33 Diagrama BPMN de *diseño de pruebas* obtenida con MOSKitt4ME

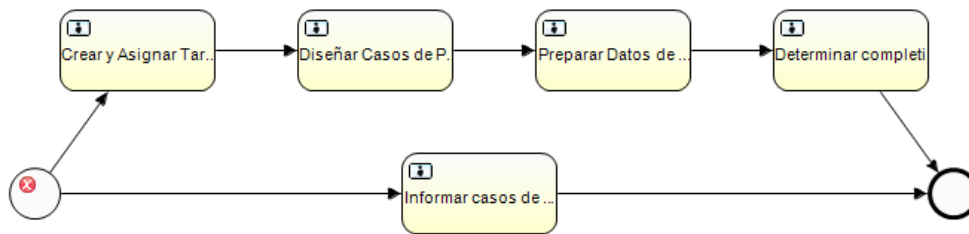
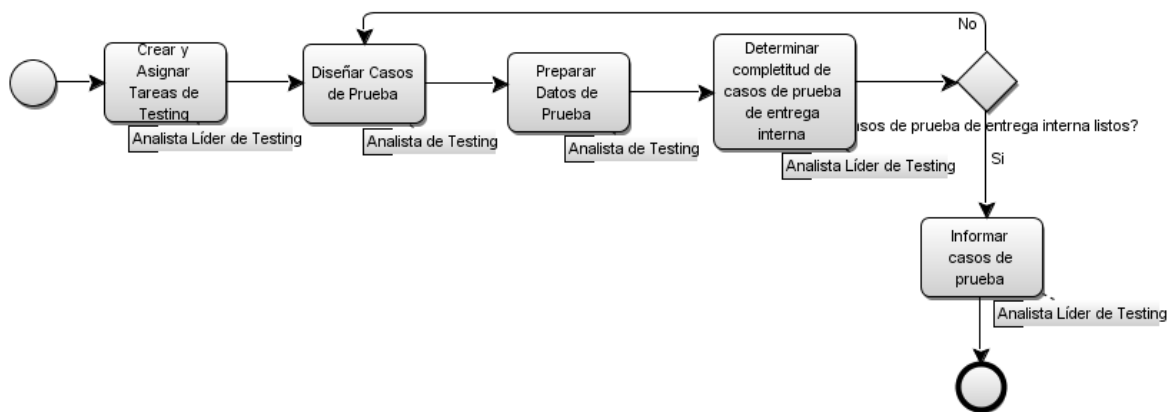


Figura 34 Diagrama BPMN de la actividad *diseño de pruebas* obtenida con la transformación implementada



4.3.8 Transformación de la actividad *Desarrollo de Solución*

Esta actividad reúne las tareas de planeación, generación de ambientes de desarrollo, implementación, pruebas unitarias, integración y en general cubre la transformación de los requerimientos en piezas de software.

La Figura 35 muestra las once tareas y un flujo compuesto por elementos de decisión, nodos *fork*, *join* y un nodo *merge*.

La Figura 36 muestra el diagrama generado con MOSKitt4ME, mientras que la Figura 37 muestra el diagrama producido con la implementación propia.

Figura 35 Diagrama de la actividad *desarrollo de solución* en EPF Composer

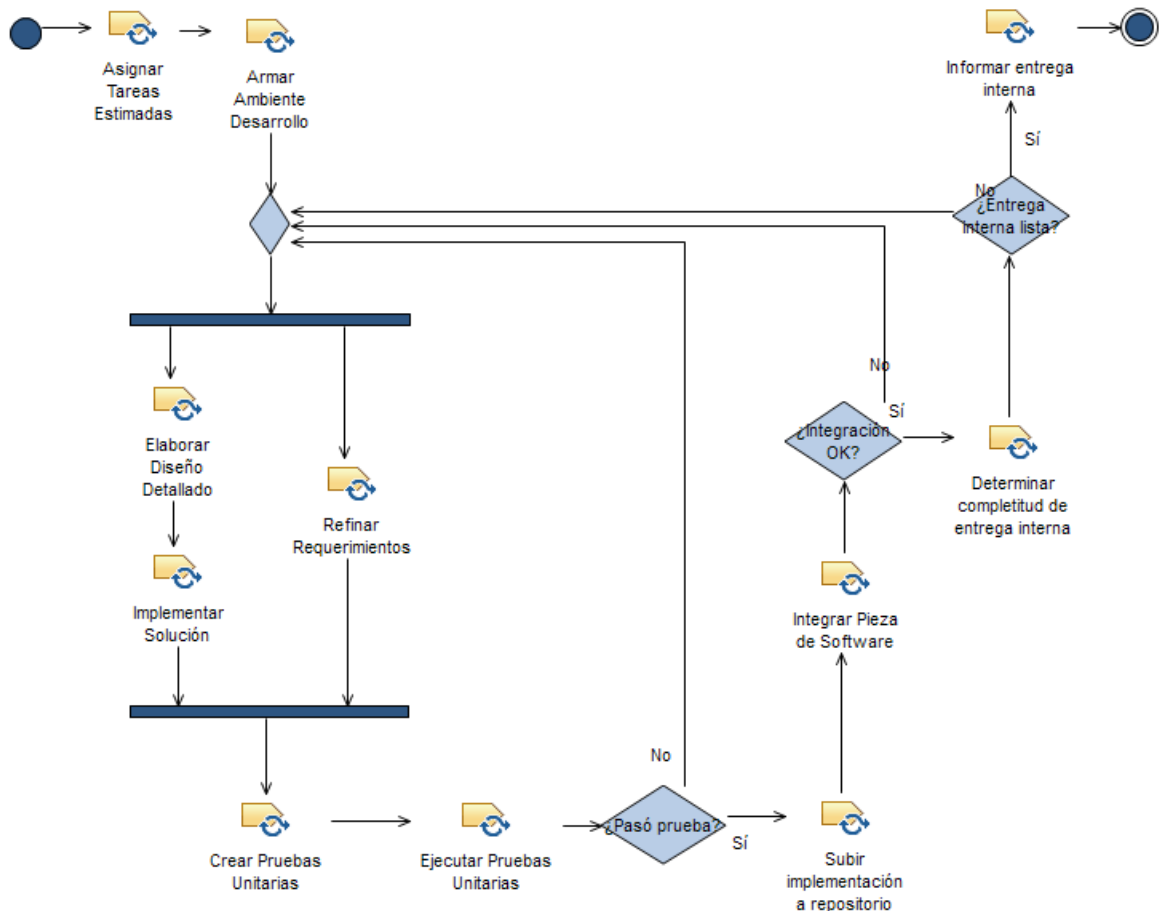


Figura 36 Diagrama BPMN de la actividad *desarrollo de solución* obtenida con MOSKitt4ME

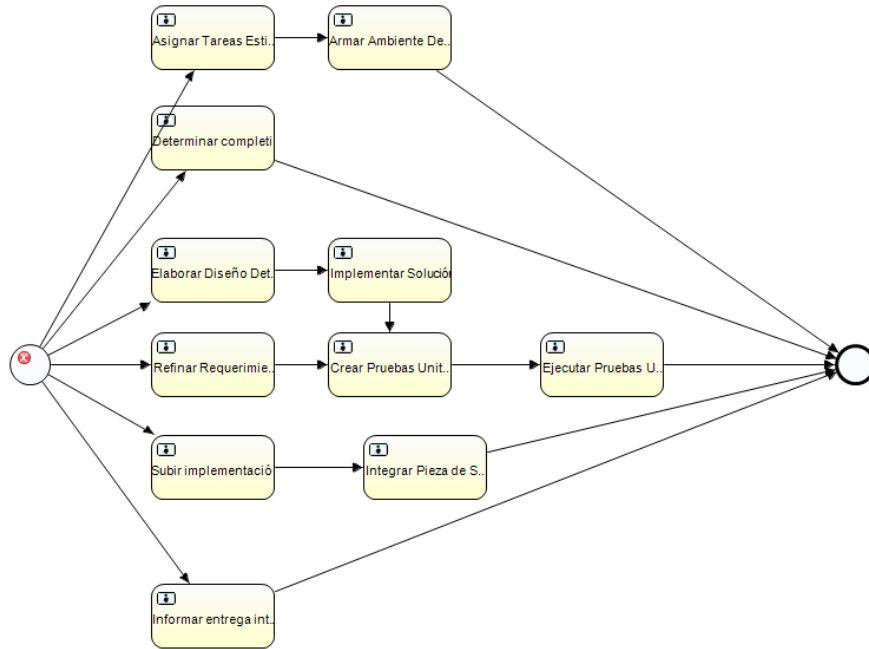
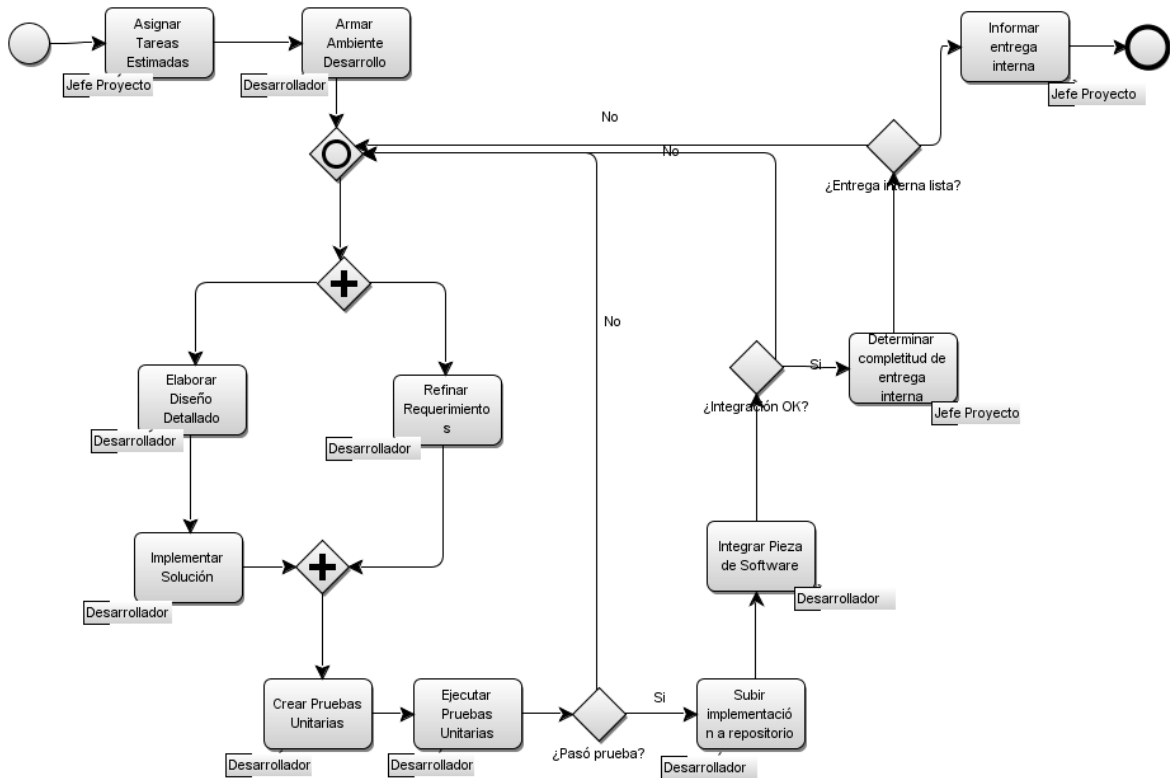


Figura 37 Diagrama BPMN de la actividad *desarrollo de solución* obtenida con la transformación implementada



4.3.9 Transformación de la actividad *Ejecución de Pruebas*

El objetivo de esta actividad es proporcionar información objetiva e independiente sobre la calidad del producto de software construido. Agrupa tres tareas: *armar ambiente de pruebas, ejecutar casos de prueba y reportar falla* que se ejecutan en un escenario que incluye tres condiciones.

El proceso fuente se presenta en la Figura 38. Los diagramas BPMN obtenidos a través de Moskkit4me y de la transformación implementada se presentan en la Figura 39 y la Figura 40 respectivamente.

Figura 38 Diagrama de la actividad *Ejecución de Pruebas* en EPF Composer

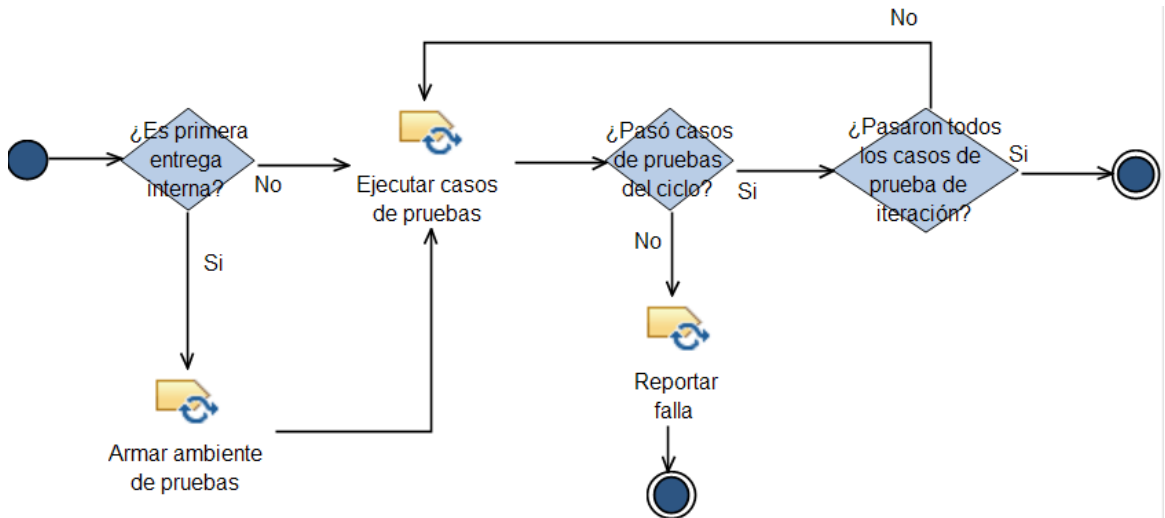


Figura 39 Diagrama BPMN de la actividad *Ejecución de Pruebas* obtenida con MOSKitt4ME

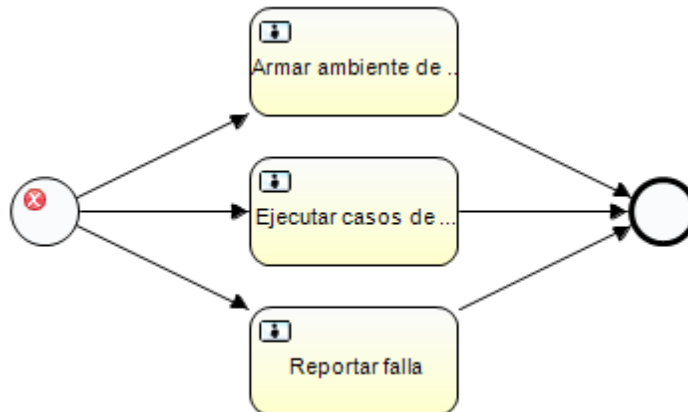
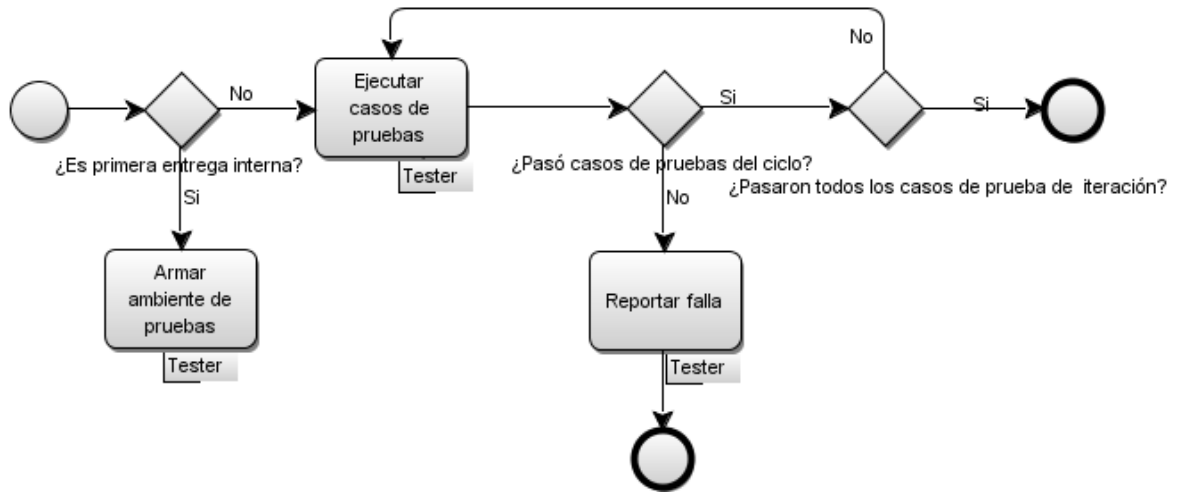


Figura 40 Diagrama BPMN de la actividad *Ejecución de Pruebas* obtenida con la transformación implementada



4.3.10 Transformación de la actividad *Liberación*

La actividad de *Liberación* consiste en la ejecución de tareas relativas a la publicación y entrega de nuevas versiones. En la Figura 41 se muestra el modelo fuente. La Figura 42 presenta el diagrama originado con MOSKitt4ME, y la Figura 43, el diagrama producido por la transformación implementada.

Figura 41 Diagrama de la actividad *Liberación* en EPF Composer

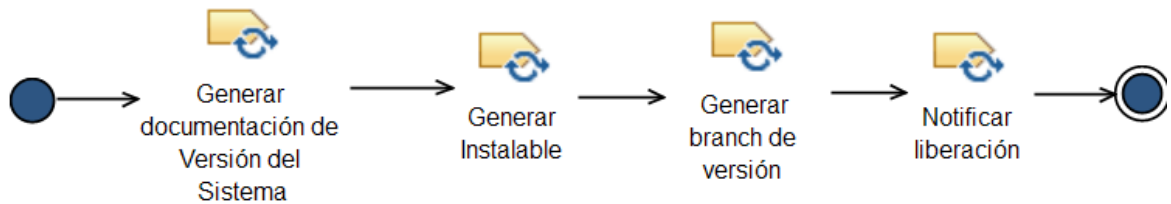


Figura 42 Diagrama BPMN de la actividad *Liberación* obtenida con MOSKitt4ME

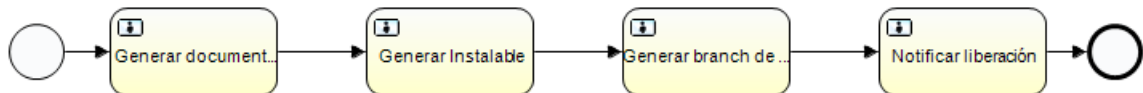
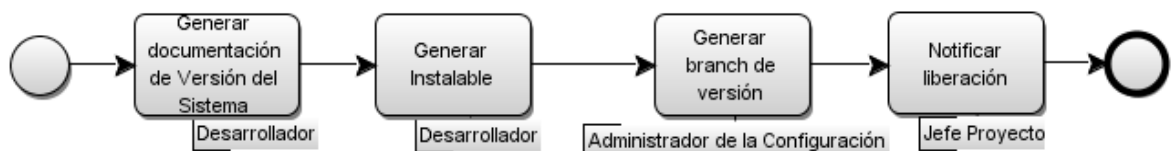


Figura 43 Diagrama BPMN de la actividad *Liberación* obtenida con la transformación implementada



4.3.11 Transformación de la actividad *Implementación*

Esta actividad está compuesta por la tarea *Planificar Entregas Internas* y cuatro actividades *Diseño Pruebas*, *Desarrollo Solución*, *Ejecución de Pruebas* y *Liberación* en un primer nivel de profundidad. Cada uno de estas actividades fue presentada en las secciones anteriores. El flujo del diagrama de implementación incluye una interesante composición de nodos *fork*, *join* y nodos de decisión, como lo ilustra la Figura 44.

EL diagrama BPMN logrado con MOSKitt4ME solamente logra transformar la tarea *Planificar Entregas Internas*, (Figura 45) mientras que el diagrama obtenido con la transformación implementada es conforme al proceso SPEM original (Figura 46)

Figura 44 Diagrama de la actividad *Implementación* en EPF Composer

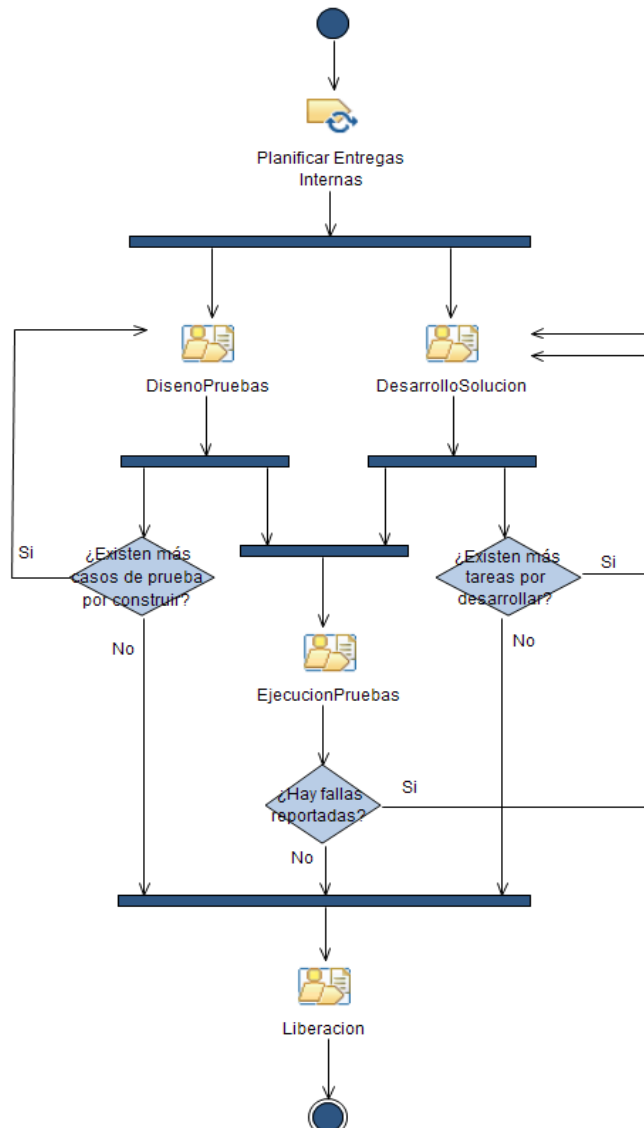


Figura 45 Diagrama BPMN de la actividad *Implementación* obtenida con MOSKitt4ME

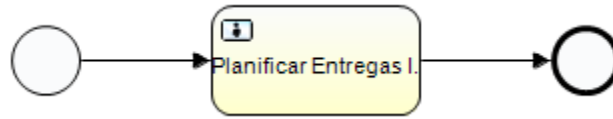
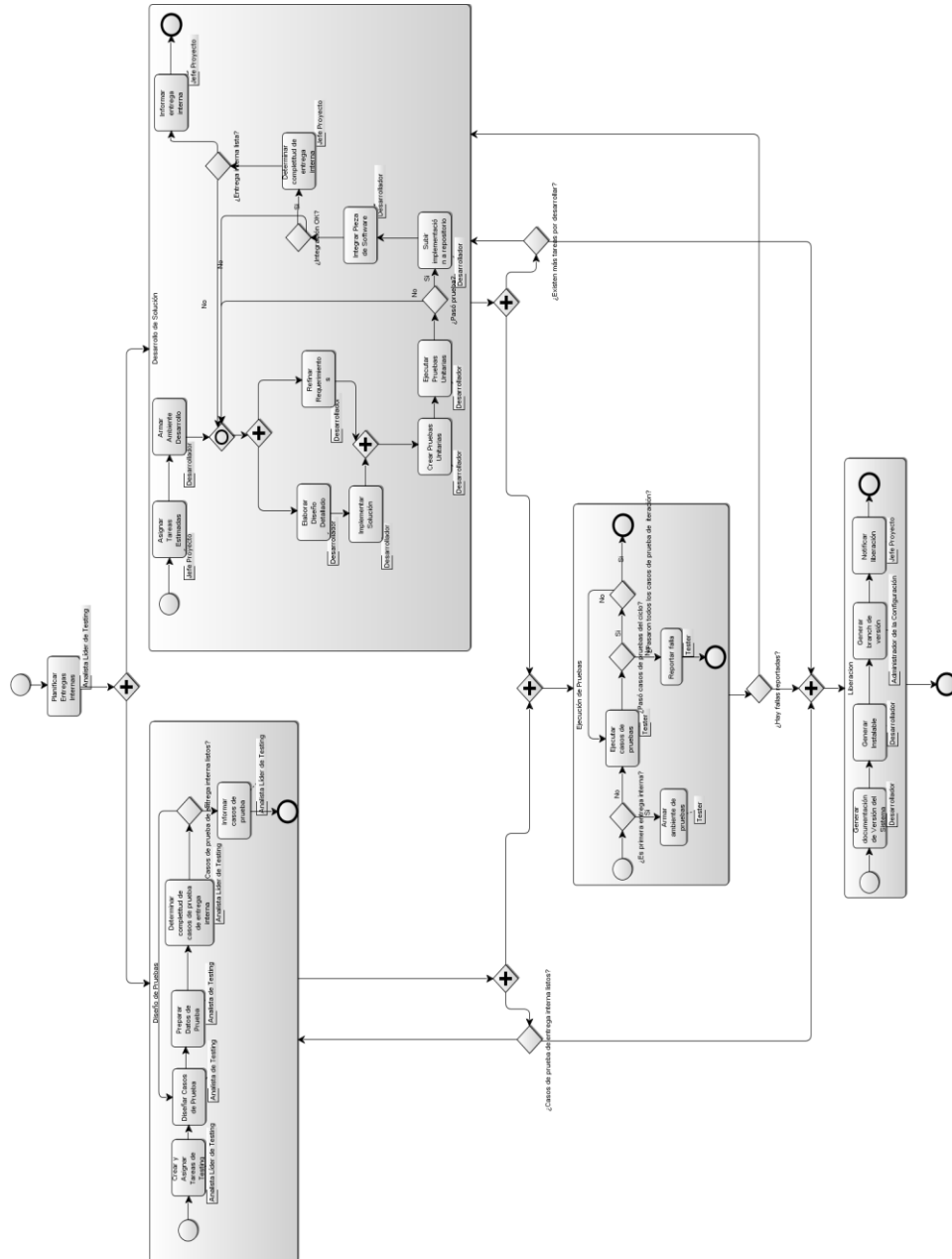


Figura 46 Diagrama BPMN de la actividad *Implementación* obtenida con la transformación implementada



Capítulo 5. Conclusiones y Recomendaciones

5.1 Resumen del trabajo realizado

En este trabajo se aplica un enfoque basado en MDE para la obtención de procesos de negocios especificados en BPMN a partir de procesos de software especificados en SPEM. Esto se logró implementando una transformación XSLT y se validó aplicándola al proceso de desarrollo de software de la empresa Chilena Mobius.

En términos técnicos, hemos sido capaces de transformar los elementos de proceso de SPEM que tienen un correspondiente en BPMN. Las estructuras *discipline activity* de SPEM se transformaron en subprocesos de BPMN, las *taskUse* de SPEM se transformaron en tareas. Los nodos de decisión y las conexiones entre tareas también son transformados conservando el contenido y el comportamiento definido en proceso de desarrollo en SPEM.

Especial atención recibieron los roles especificados en SPEM. Las tareas del proceso BPMN resultante están rotuladas con el rol que debe desempeñarlas. Esto ayuda a la usabilidad y aplicabilidad de la solución, sin transgredir el estándar BPMN.

Se propone un metamodelo capaz de instanciarse en modelos de recursos humanos para la representación de roles, recursos y tareas.

La transformación de las *discipline activity* de SPEM a subprocesos de BPMN fue una decisión de compromiso: o bien organizábamos el BPMN en *pools y lanes* distribuyendo las tareas de acuerdo con los respectivos roles, o bien las agrupábamos por subprocesos. Se optó por la segunda opción ya que si bien ambas vistas eran posibles, no se pueden obtener de forma simultánea.

La transformación fue capaz de procesar satisfactoriamente la totalidad de actividades del proceso de desarrollo de una empresa real. Los resultados obtenidos a través de nuestra transformación fueron comparados con los que se pueden obtener con MOSKitt4ME, la herramienta más nueva y poderosa del mercado que tiene los mismos

objetivos: transformar de SPEM a BPMN. Mostramos con el caso de estudio en la empresa real que nuestra transformación resulta mucho más expresiva y usable.

5.2 Revisión de Objetivos

A continuación se relacionan los objetivos específicos propuestos y se realiza un análisis acerca de su cubrimiento.

- *Definir un metamodelo para la especificación de estructuras de recursos humanos en procesos de desarrollo de software.*

Este objetivo se cubre enteramente a través de la especificación del metamodelo planteado en la sección 3.2 que permite la representación de las estructuras rol, recurso y tarea. Si bien no se usa aún en todo su potencial el modelo de recursos humanos, ya sirve para modelar equipos de trabajo para la ejecución del proyecto.

- *Definir una transformación que tome como entrada un proceso conforme con el metamodelo UMA y un modelo de recursos humanos conforme con el metamodelo definido en el punto anterior, y produzca como salida un proceso conforme con el metamodelo BPMN y que incluya los recursos humanos concretos que participarán del proyecto.*

Este objetivo se cubre parcialmente. Se implementa una transformación de UMA/SPEM que permite obtener diagramas BPMN representando las estructuras de procesos de desarrollo y manteniendo la consistencia del flujo y comportamiento del modelo fuente. El diagrama BPMN generado no incluye los recursos humanos concretos pero si los roles responsables de la ejecución de cada tarea conforme al proceso de desarrollo origen.

- *Establecer las herramientas computacionales que soporten las transformaciones entre modelos.*

Este objetivo se cumple enteramente con la especificación del archivo de transformación XSLT que se ejecuta en cualquier navegador *Firefox* (a través del motor Saxon-CE). Como el modelo resultante conforma con el estándar BPMN 2.0, se pudo escoger para su visualización entre una serie de herramientas disponibles. Finalmente se usó *Yaoqiang BPMN Editor* para esta visualización.

- Validar la estrategia de transformación a modelos BPMN en al menos dos proyectos de software de una PyME.

Este objetivo se cumple con la transformación satisfactoria de la totalidad de actividades del proceso de desarrollo de una empresa real. Como en el diagrama BPMN generado se incluyen los roles responsables de la ejecución de cada tarea y no instancias de recursos humanos completos, se resolvió omitir la transformación de procesos de proyectos.

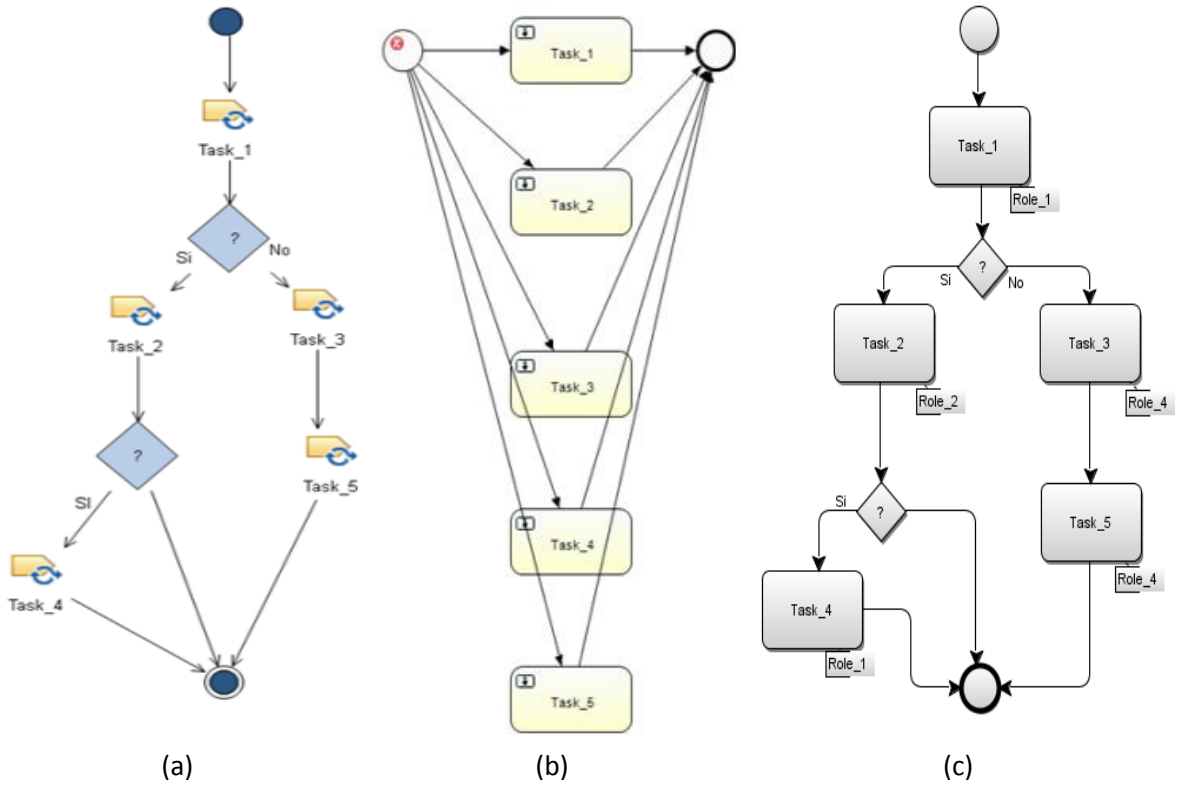
5.3 Principales Contribuciones

Como aportes principales al área de modelamiento de procesos de desarrollo de software se pueden señalar los siguientes:

1. Se establece el estado del arte actualizado de los trabajos en el área.
2. Se logra la transformación de nodos *fork*, *join*, *merge* y de decisión para conservar el flujo del proceso original.
3. Se logra mayor visibilidad del proceso de negocio en BPMN al agregar los roles como etiquetas de las respectivas tareas.
4. Los modelos generados con la transformación mantienen consistencia con lo establecido en el estándar BPMN 2.0.

Estas condiciones no se cumplían con el uso de MOSKitt4ME, la herramienta existente para tal propósito. La Figura 47 (a) ilustra un proceso sencillo y las representaciones BPMN obtenidas con el uso de MOSKitt4ME (b) y con la transformación implementada en el presente trabajo (c). Aquí queda de manifiesto el logro de las contribuciones 2, 3 y 4.

Figura 47 Comparación de Diagramas BPMN obtenidos



5.4 Lecciones Aprendidas

Respecto a la elección de XSLT como lenguaje de transformación, si bien es cierto lo advertido en [49] en cuanto a las restricciones de modularidad, eficiencia, reusabilidad y mantenimiento, encontramos que las capacidades ofrecidas por el lenguaje fueron suficientes para los objetivos propuestos.

La persistencia de los datos en EPF Composer es débil. A menudo surgen serias inconsistencias entre el contenido del `model.xml` y el contenido del `diagram.xml`. Estas inconsistencias hacen que la recuperación de las entidades en la ejecución de la transformación resulte vacía y obliga a realizar nuevamente la especificación de cada actividad del proceso de desarrollo en EPF Composer.

La organización de los diagramas BPMN se puede realizar desde dos perspectivas: de un lado, haciendo uso de *pools* y *lanes* distribuyendo las tareas de acuerdo con los respectivos roles, o bien agrupándolas por subprocessos; en este trabajo se optó por la segunda opción. Si bien ambas vistas eran posibles, no se pueden obtener de forma simultánea.

5.5 Trabajo Futuro

El desarrollo del presente trabajo permite la formulación de diferentes propuestas relacionadas con el problema de investigación formulado al inicio.

La usabilidad de la transformación implementada en el aún no ha sido validada con usuarios finales. Validar la usabilidad de modo empírico permitirá obtener medidas respecto a la adopción potencial de la transformación como una herramienta para la obtención de procesos de negocio a partir de procesos de desarrollo en entornos industriales.

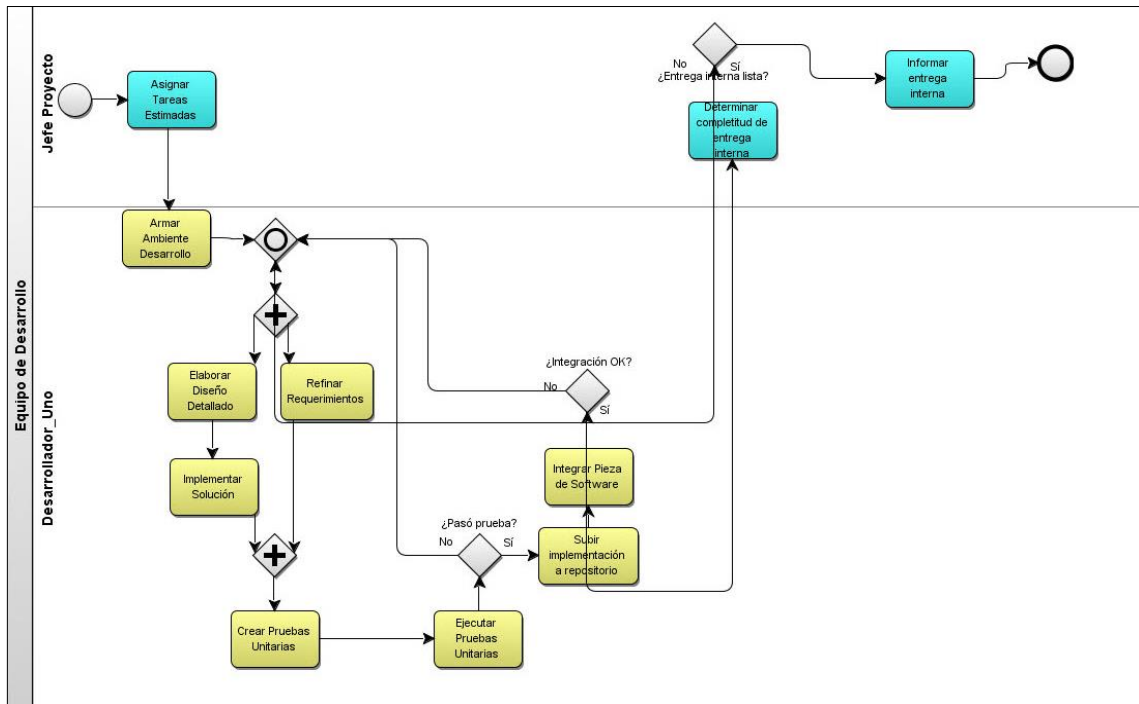
Otro escenario de validación que permitirá la contrastación empírica de las hipótesis subyacentes del proyecto será la ejecución de los procesos de negocio obtenidos a través de la transformación implementada. Para esto se requiere de un conjunto de parámetros relativos a la automatización y el control del proceso.

Una actividad a abordar será la elaboración de un producto a partir de la compilación de los elementos producidos como parte de este proyecto. El producto busca la integración de la transformación como un *plugin* de la plataforma Eclipse, donde se componga con EPF Composer y con un *plugin* con capacidad de despliegue y edición de BPMN 2.0.

El uso de etiquetas permite la asociación de las tareas con sus respectivos roles y se presenta como una alternativa para agregar mayor información en el diagrama. Sin embargo la representación de recursos humanos concretos en BPMN permitiría una administración de estos a través de plataformas de *BPMS* que despache las tareas a las personas responsables de las mismas.

Como un nuevo reto en el proyecto se ha considerado mapear los roles participantes en el proceso de desarrollo a los correspondientes *pools* y *lanes* en el proceso de negocio. Este mapeo permite obtener diagramas BPMN como el presentado en la Figura 48 que ilustra la actividad Desarrollo Solución.

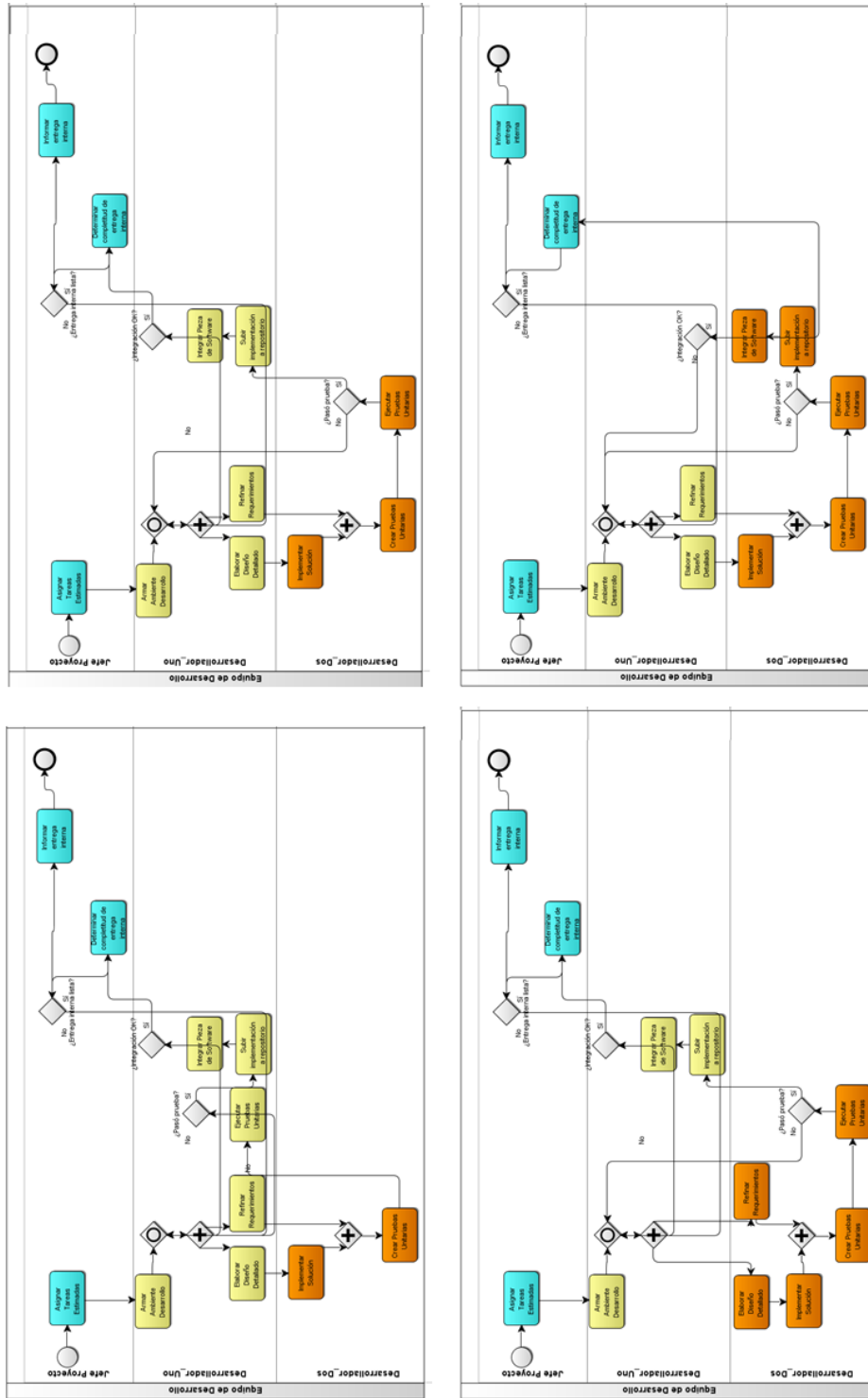
Figura 48 Diagrama BPMN actividad *Desarrollo Solución* con pool y lanes



Dado que es posible tener más de un recurso desempeñando un mismo rol y bajo el supuesto de que tienen las mismas habilidades que les permitan ejecutar las tareas asignadas a su rol, la asignación más apropiada de tareas puede encontrarse como un problema de investigación relevante.

Así, en el caso de la transformación de la disciplina Desarrollo Solución del proceso de Mobius, cuando se tienen dos recursos humanos en el desarrollo es posible generar diferentes asignaciones, lo que derivaría en obtener diferentes modelos en BPMN. Para ilustrar mejor esta idea, algunos de los modelos posibles se muestran en la Figura 49. Esta situación puede entenderse como un problema de asignación clásico, en donde se dispone de algunos recursos (máquinas o personas) para la elaboración de ciertos productos a costo mínimo. Esto es aplicable en variados contextos como la designación de tareas a empleados, de territorios a vendedores, de contratos a postores o de trabajos a plantas. Tradicionalmente ha sido abordado desde la perspectiva de optimización combinatoria en el área de investigación de operaciones. Otras soluciones se originan al aplicar técnicas de simulación, técnicas de aprendizaje maquina tales como las redes neuronales, máquinas de soporte vectorial, y algoritmos genéticos, entre otros.

Figura 49 Escenarios de asignación de tareas con dos desarrolladores



A. Código fuente de la transformación

La transformación implementada en XSLT y los recursos referentes a su uso se encuentran disponibles siguiendo el enlace: adapte.dcc.uchile.cl:8080/transformationSPeM2BPMN/. A continuación y a modo ilustrativo se presenta el código fuente completo de la transformación. Con la presente versión se obtuvieron cada uno de los modelos BPMN presentados en el capítulo de validación.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
    xmlns:notation="http://www.eclipse.org/gmf/runtime/1.0.2/notation"
    xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:org.eclipse.epf.uma="http://www.eclipse.org/epf/uma/1.0.6/uma.ecore">

<xsl:template name="outgoingtokenize">
  <xsl:param name="text" select="."/>
  <xsl:param name="separator" select="' '>
  <xsl:choose>
    <xsl:when test="not(contains($text, $separator))">
      <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
select="normalize-space($text)"/></outgoing>
    </xsl:when>
    <xsl:otherwise>
      <outgoing xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
select="normalize-space(substring-before($text, $separator))"/></outgoing>
      <xsl:call-template name="outgoingtokenize">
        <xsl:with-param name="text" select="substring-after($text, $separator)"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template name="incomingtokenize">
  <xsl:param name="text" select="."/>
  <xsl:param name="separator" select="' '>
  <xsl:choose>
    <xsl:when test="not(contains($text, $separator))">
      <incoming xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
select="normalize-space($text)"/></incoming>
    </xsl:when>
    <xsl:otherwise>
      <incoming xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
select="normalize-space(substring-before($text, $separator))"/></incoming>
      <xsl:call-template name="incomingtokenize">
        <xsl:with-param name="text" select="substring-after($text, $separator)"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

```

        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template name="procesarElementos">
    <xsl:for-each select="node">
        <xsl:choose>
            <!-- Escribir nodos de inicio -->
            <xsl:when test="@xmi:type = 'uml:InitialNode'">
                <xsl:variable name="idInitialNote" select="@xmi:id"/>
                <xsl:variable name="outgoingInitialNote" select="@outgoing"
/>
                <startEvent
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
                    <xsl:attribute name="id">
                        <xsl:value-of select="@xmi:id" />
                    </xsl:attribute>
                    <xsl:if test="string-length(@outgoing) > 0">
                        <xsl:call-template name="outgoingtokenize">
                            <xsl:with-param name="text"
select="@outgoing"/>
                            <xsl:with-param name="separator"
select="' '"/>
                        </xsl:call-template>
                    </xsl:if>
                </startEvent>
            </xsl:when>
            <!-- Escribir nodos de fin -->
            <xsl:when test="@xmi:type = 'uml:ActivityFinalNode'">
                <endEvent
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
                    <xsl:attribute name="id">
                        <xsl:value-of select="@xmi:id" />
                    </xsl:attribute>
                    <xsl:if test="string-length(@incoming) > 0">
                        <xsl:call-template name="incomingtokenize">
                            <xsl:with-param name="text"
select="@incoming"/>
                            <xsl:with-param name="separator" select="' '"/>
                        </xsl:call-template>
                    </xsl:if>
                </endEvent>
            </xsl:when>
            <!-- Escribir nodos de decision -->
            <xsl:when test="@xmi:type = 'uml:DecisionNode'">
                <exclusiveGateway
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
                    gatewayDirection="Diverging">
                    <xsl:attribute name="id"><xsl:value-of
select="@xmi:id" /></xsl:attribute>
                    <xsl:attribute name="name"><xsl:value-of
select="@name" /></xsl:attribute>
                    <xsl:if test="string-length(@incoming) != 0">
                        <xsl:call-template name="incomingtokenize">
                            <xsl:with-param name="text"
select="@incoming"/>
                            <xsl:with-param name="separator" select="' '"/>
                        </xsl:call-template>
                    </xsl:if>
                    <xsl:if test="string-length(@outgoing) != 0">
                        <xsl:call-template name="outgoingtokenize">
                            <xsl:with-param name="text"
select="@outgoing"/>
                            <xsl:with-param name="separator" select="' '"/>
                        </xsl:call-template>
                    </xsl:if>
                </exclusiveGateway>

```

```

        </xsl:when>
        <!-- Escribir fork & join -->
        <xsl:when test="(@xmi:type = 'uml:ForkNode') or (@xmi:type =
'uml:JoinNode') ">
            <parallelGateway
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
                <xsl:choose>
                    <xsl:when test="@xmi:type = 'uml:ForkNode'">
                        <xsl:attribute
name="gatewayDirection">Diverging</xsl:attribute>
                    </xsl:when>
                    <xsl:when test="@xmi:type = 'uml:JoinNode'">
                        <xsl:attribute
name="gatewayDirection">Converging</xsl:attribute>
                    </xsl:when>
                </xsl:choose>

                <xsl:attribute name="name"><xsl:value-of select="@name"
/></xsl:attribute>

                <xsl:attribute name="id"><xsl:value-of
select="@xmi:id" /></xsl:attribute>

                <xsl:if test="string-length(@incoming) != 0">
                    <xsl:call-template name="incomingtokenize">
                        <xsl:with-param name="text"
select="@incoming" />

                        <xsl:with-param name="separator" select="' '"/>
                    </xsl:call-template>
                </xsl:if>
                <xsl:if test="string-length(@outgoing) != 0">
                    <xsl:call-template name="outgoingtokenize">
                        <xsl:with-param name="text"
select="@outgoing" />

                        <xsl:with-param name="separator" select="' '"/>
                    </xsl:call-template>
                </xsl:if>
            </parallelGateway>
        </xsl:when>
        <!-- Escribir nodo de mezcla -->
        <xsl:when test="@xmi:type = 'uml:MergeNode'">
            <parallelGateway
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
                <xsl:attribute
name="gatewayDirection">Converging</xsl:attribute>

                <xsl:attribute name="name"><xsl:value-of select="@name"
/></xsl:attribute>

                <xsl:attribute name="id"><xsl:value-of
select="@xmi:id" /></xsl:attribute>

                <xsl:if test="string-length(@incoming) != 0">
                    <xsl:call-template name="incomingtokenize">
                        <xsl:with-param name="text"
select="@incoming" />

                        <xsl:with-param name="separator" select="' '"/>
                    </xsl:call-template>
                </xsl:if>
                <xsl:if test="string-length(@outgoing) != 0">
                    <xsl:call-template name="outgoingtokenize">
                        <xsl:with-param name="text"
select="@outgoing" />

                        <xsl:with-param name="separator" select="' '"/>
                    </xsl:call-template>
                </xsl:if>
            </parallelGateway>
        </xsl:when>

        <xsl:when test="@xmi:type = 'uml:ActivityParameterNode'" >
            <xsl:variable name="name" select="@name"/>
            <xsl:variable name="id" select="@xmi:id"/>

```

```

        <task xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
            completionQuantity="1"
            isForCompensation="false"
            startQuantity="1">
            <xsl:attribute name="id"><xsl:value-of select="$id"
/></xsl:attribute>
            <xsl:attribute name="name"><xsl:value-of
select="$name" /></xsl:attribute>
            <xsl:if test="string-length(@incoming) != 0">
            <xsl:call-template name="incomingtokenize">
            <xsl:with-param name="text"
select="@incoming"/>
            <xsl:with-param name="separator" select="'
'"/>
            </xsl:call-template>
            </xsl:if>
            <xsl:if test="string-length(@outgoing) != 0">
            <xsl:call-template name="outgoingtokenize">
            <xsl:with-param name="text" select="@outgoing"/>
            <xsl:with-param name="separator" select="'
'"/>
            </xsl:call-template>
            </xsl:if>
        </task>
    </xsl:when>
</xsl:choose>
</xsl:for-each>
<xsl:for-each select="edge">
    <!-- Escribir flechas -->
    <xsl:choose>
        <xsl:when test="@xmi:type = 'uml:ControlFlow'" >
            <xsl:if test="(string-length(@source) != 0) and (string-
length(@target) != 0)">
                <sequenceFlow
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
                    <xsl:attribute name="id"><xsl:value-of
select="@xmi:id" /></xsl:attribute>
                    <xsl:attribute name="name"><xsl:value-of
select="@name" /></xsl:attribute>
                    <xsl:attribute name="sourceRef"><xsl:value-of
select="@source" /></xsl:attribute>
                    <xsl:attribute name="targetRef"><xsl:value-of
select="@target" /></xsl:attribute>
                </sequenceFlow>
            </xsl:if>
        </xsl:when>
    </xsl:choose>
</xsl:for-each>
</xsl:template>
<xsl:template name="procesarNotas">
    <xsl:for-each select="node">
        <xsl:choose>
            <xsl:when test="@xmi:type = 'uml:ActivityParameterNode'" >
                <xsl:variable name="name" select="@name"/>
                <xsl:variable name="id" select="@xmi:id"/>
                <!-- Buscar los roles a los que está asociado la tarea y
genera una nota -->
                <xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/processElements">
                    <xsl:if test="( @xsi:type =
'org.eclipse.epf.uma:TaskDescriptor') and (@presentationName = $name)">
                        <xsl:variable name="idRol"
select="@performedPrimarilyBy"/>
                        <xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/processElements">
                            <xsl:if

```

```

test="(@xsi:type='org.eclipse.epf.uma:RoleDescriptor') and (@xmi:id = $idRol)"
                                <textAnnotation
textFormat="text/plain"

    xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
                                <xsl:attribute
name="id"><xsl:value-of select="concat('N_', $id)" /></xsl:attribute>
                                <text
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
select="@presentationName" /></text>
                                </textAnnotation>
                                <association
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
associationDirection="None">
                                <xsl:attribute
name="id"><xsl:value-of select="concat('A_', $id)" /></xsl:attribute>
                                <xsl:attribute
name="sourceRef"><xsl:value-of select="concat('N_', $id)" /></xsl:attribute>
                                <xsl:attribute
name="targetRef"><xsl:value-of select="$id" /></xsl:attribute>
                                </association>

                                </xsl:if>
                            </xsl:for-each>

                        </xsl:if>
                    </xsl:for-each>

                </xsl:when>
            </xsl:choose>
        </xsl:for-each>
    </xsl:template>

<xsl:template name="procesarNotasChild">
    <xsl:for-each select="node">
        <xsl:choose>
            <xsl:when test="@xmi:type = 'uml:ActivityParameterNode'" >
                <xsl:variable name="name" select="@name"/>
                <xsl:variable name="id" select="@xmi:id"/>

                <!-- Buscar los roles a los que está asociado la tarea y
genera una nota -->
                <xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/childPackages/processEleme
nts">
                    <xsl:if test="(@xsi:type =
'org.eclipse.epf.uma:TaskDescriptor') and (@presentationName = $name)">
                        <xsl:variable name="idRol"
select="@performedPrimarilyBy"/>

                        <xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/childPackages/processEleme
nts">
                            <xsl:if
test="(@xsi:type='org.eclipse.epf.uma:RoleDescriptor') and (@xmi:id = $idRol)"
                                <textAnnotation
textFormat="text/plain"

                                xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
                                <xsl:attribute
name="id"><xsl:value-of select="concat('N_', $id)" /></xsl:attribute>
                                <text
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"><xsl:value-of
select="@presentationName" /></text>
                                </textAnnotation>
                                <association
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

```

```

associationDirection="None">
name="id"><xsl:value-of select="concat('A_', $id)" /></xsl:attribute>
name="sourceRef"><xsl:value-of select="concat('N_', $id)" /></xsl:attribute>
name="targetRef"><xsl:value-of select="$id" /></xsl:attribute>
</association>
</xsl:if>
</xsl:for-each>
</xsl:if>
</xsl:for-each>
</xsl:when>
</xsl:choose>
</xsl:for-each>
</xsl:template>
<xsl:template match="/root/diagram">
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:tns="http://sourceforge.net/bpmn/definitions/_1385326020117"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
expressionLanguage="http://www.w3.org/1999/XPath"
id="_1385326020117"
name="CMMM"
targetNamespace="http://sourceforge.net/bpmn/definitions/_1385326020117"
typeLanguage="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
http://bpmn.sourceforge.net/schemas/BPMN20.xsd">
<!-- Proceso Padre que contiene las demás tareas -->
<process isClosed="false"
isExecutable="true"
processType="None">
<!-- Buscar la actividad padre -->
<xsl:for-each select="xmi:XMI/uml:Activity">
<!-- Procesar que esta en la raíz. No se procesan actividades -->
<xsl:if test="position() = 1">
<xsl:call-template name="procesarElementos" />
</xsl:if>
<!-- Procesar tag que está en un sub proceso -->
<xsl:if test="position() != 1">
<subProcess>
<!-- Aquí se debe buscar el id en los
uml:StructuredActivityNode -->
<xsl:variable name="nameActivity" select="@name"/>
<!-- Definir nombre -->
<xsl:attribute name="name">
<xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/childPackages/processElements">
<xsl:if
test="( @xsi:type='org.eclipse.epf.uma:Activity' or
@xsi:type='org.eclipse.epf.uma:CapabilityPattern' ) and @name = $nameActivity">
<xsl:value-of
select="@presentationName" />
</xsl:if>
</xsl:for-each>

```



```

        </xsl:attribute>
        <!-- Definir id -->
        <xsl:attribute name="id">
            <xsl:for-each
select="/root/model/xmi:XMI/org.eclipse.epf.uma:ProcessComponent/childPackages/processElements">
                <xsl:if
test="(@xsi:type='org.eclipse.epf.uma:Activity' or
@xsi:type='org.eclipse.epf.uma:CapabilityPattern') and @name = $nameActividad">
                    <xsl:variable
name="nameActividad" select="@presentationName" />
                    <xsl:for-each
select="/root/diagram/xmi:XMI/uml:Activity/node">
                        <xsl:if
test="@xmi:type='uml:StructuredActivityNode' and @name = $nameActividad">
                            <xsl:value-of
select="@xmi:id" />
                                </xsl:if>
                            </xsl:for-each>
                        </xsl:if>
                    </xsl:for-each>
                </xsl:attribute>
                <xsl:call-template name="procesarElementos" />
                <xsl:call-template name="procesarNotasChild" />
            </subProcess>
        </xsl:if>
    </xsl:for-each>
    <xsl:for-each select="xmi:XMI/uml:Activity">
        <xsl:call-template name="procesarNotas" />
    </xsl:for-each>
</process>
</definitions>
</xsl:template>
</xsl:stylesheet>
```


B. Bibliografía

- [1] **Feiler, P. H. & Humphrey, W. S.,** *Software Process Development and Enactment: Concepts and Definitions.* International Conference of Software Process (ICSP),1993. IEEE Computer Society, p 28- 40.
- [2] **Object Management Group,** *Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0;* Formal specification; formal/2008-04-01; April 2008, <http://www.omg.org/spec/SPEM/2.0/>,
- [3] **Hammer, M. & Champy, J.,** *Reengineering the Corporation.* New York; HarperCollins, (1993).
- [4] **Smith, H., Neal, D., Ferrara, L., Hayden, F.,** *The Emergence of Business Process Management,* Computer Sciences Corporation Research Services, Enero (2002).
- [5] **Object Management Group,** *Business Process Model and Notation, Version 2.0;* Formal specification; formal/2011-01-03; January 2011, <http://www.omg.org/spec/BPMN/2.0/PDF/>, accessed April 2013.
- [6] **Object Management Group,** *Unified Modeling Language Superstructure Specification,* version 2.1.1. Document formal/2007-02-05, February 2007. <http://www.omg.org/cgi-bin/doc?formal/2007-02-05>, accessed April 2013.
- [7] «**BPMN FAQ**». <http://www.omg.org/bpmn/Documents/FAQ.htm> accessed April 2013.
- [8] **Workflow Management Coalition,** *Workflow Standard - Workflow Process Definition Interface -XML Process Definition Language, Version 2.2;* WFMC-TC-1025, August 2012, [http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20\(2012-08-30\).pdf/](http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf/), accessed April 2013.

- [9] **OASIS**, *Web Services Business Process Execution Language Version 2.0. wsbpel-primer*; May 2007 <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>; /, accessed April 2013.
- [10] **Object Management Group**, *Model Driven Architecture (MDA) Guide Version 1.0*; May 2003, http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf, accessed April 2013.
- [11] **MOLA Project**. *The MOLA Language Reference Manual Version 2.0 final*; December, 2007; <http://mola.mii.lu.lv/>, accessed April 2013.
- [12] **Kolovos, D. S., Paige, R. F. and Polack F. A. C.** *The Epsilon Transformation Language (ETL)*. In Proceedings of Theory and Practice of Model Transformations: First International Conference, ICMT 2008, Zürich, Switzerland, July 1-2, 2008
- [13] **Jouault F., Kurtev I.** *Transforming Models with ATL*. Workshop in Model Transformation in Practice at the MODELS 2005 Conference. Montego Bay, Jamaica, Oct 3, 2005
- [14] **Object Management Group**, *Meta Object Facility (MOF) 2. Query/View/Transformation (QVT) Formal specification*; formal/2011-01-01, January 2011, <http://http://www.omg.org/spec/QVT/1.1/PDF/>, accessed April 2013.
- [15] **Chan, D., Leung K. R.P.H.**, *Software Development as a Workflow Process* APSEC, p. 282-291, Fourth Asia-Pacific Software Engineering and International Computer Science Conference (APSEC'97 / ICSC'97), IEEE 1997.
- [16] **Min, F., Jin, Y., Minghui, W.**, *A Framework of Workflow-based Software Development Process Supported Plataform, Computer Supported Cooperative Work in Design*, 2004. Proceedings. The 8th International Conference on 26 May 2004.
- [17] **Gardner, T., Amsden, J., Griffin, C., Iyengar, S.** *Draft UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0. Version 1.1*. IBM. June 9th 2003. http://www.ibm.com/developerworks/rational/library/content/04April/3103/3103_UMLProfileForBusinessProcesses1.1.pdf. /, accessed April 2013
- [18] **Gallina, B., Guelfi, N., Mammari, A.** *Structuring Business Nested Processes Using UML 2.0 Activity Diagrams and Translating into XPDL*. Proceedings of the 3rd GI-

- Workshop XML4BPM - XML Integration and Transformation for Business Process Management. Passau, Germany, February 2006. Pages: 281 - 296.
- [19] **Kalnins, A., Vitolins, V.** *Use of UML and Model Transformations for Workflow Process Definitions Databases and Information Systems*, BalticDB&IS'2006, edited by Olegas Vasilecas, Johann Eder, Albertas Caplinskas, Vilnius, Technika, 2006, pp. 3.-15
- [20] **Filograna, A., Giunta, G., Ingraffia, N., Loiacono, L.** *An integrated approach for modelling Business Processes using BPMN and XPDL standards*, 2007. <http://semantics.eng.it/bxmodeller/docs/An%20integrated%20approach%20for%20modelling%20business%20processes%20using%20BPMN%20and%20XPDL%20standards.pdf> /, accessed April 2013
- [21] **Argañaraz, M., Funes, A., Dasso, A.** *An MDA Approach to Business Process Model Transformations in "Software Engineering in Argentina: present and future trends. Extended version of selected papers ASSE 2009"*, Electronic Journal SADIO (EJS), Volumen 9, Numero 1, 2010
- [22] **France Telecom R & D:** SmartQVT Documentation. <http://smartqvt.elibel.tm.fr> último acceso Abril de 2013.
- [23] **M. Cervera, M. Albert, V. Torres, V. Pelechano.** *A Model-Driven Approach for the Design and Implementation of Software Development Methods*. International Journal of Information System Modeling and Design (IJISMD), vol. 3(4), pp. 86-103, 2012
- [24] **Awad, A., Grosskopf, A., Meyer, A., and Weske, M.** *Enabling resource assignment constraints in BPMN*. Technical report, BPT, Hasso Plattner Institute, 2009.
- [25] **Link, S., Hoyer, P., Schuster T., and Abeck, S.** *Model-Driven Development of Human Tasks for Workflows* in 2008 The Third International Conference on Software Engineering Advances. IEEE, pp. 329-335, 2008
- [26] **Agrawal, A., Amend, M., et al.:** *Web Services Human Task (WS-HumanTask)*, version 1.0, June 2007. http://incubator.apache.org/hise/WS-HumanTask_v1.pdf accessed April 2013
- [27] **Stroppi, L., Chiotti, O., Villarreal, P.** *Extending BPMN 2.0: Method and Tool Support*. III International Workshop on the Business Process Model and Notation (BPMN 2011). Switzerland, 2011. Lecture Notes in Business

- Information Processing (LNBIP), Vol 95, pp 59-73, Springer-Verlag Berlin Heidelberg 2011.
- [28] **Russell, A. ter Hofstede et al.** *Workflow Resource Patterns*. BETA Working Paper Series, WP 127, Eindhoven University of Technology, 2004.
- [29] **Geambasu C.V.**, *BPMN vs. UML Activity Diagram for Business Process Modeling*, *Journal of Accounting and Management Information Systems*, 11 (4), pp. 637-651 2012
- [30] **Kleppe A., Warmer J., Bast W.**, *MDA Explained, The Model Driven Architecture: Practice and Promise*. Addison-Wesley, Boston, 2008.
- [31] **Object Management Group**, *OMG MOF 2 XMI Mapping Specification, Version 2.4.1*. Document Formal specification; formal/2011-01-03; June 2013. <http://www.omg.org/spec/XMI/2.4.1/PDF/>, accessed November 2013.
- [32] **IBM Support Portal**, *Compatibility between IBM Rational Method Composer and EPFC*, June 2009. <http://www-01.ibm.com/support/docview.wss?uid=swg21389689>, accessed November 2013.
- [33] **Eclipse Foundation**, *Eclipse Process Framework (EPF) Composer 1.0 Architecture Overview*, April 2007. http://www.eclipse.org/epf/composer_architecture/, accessed November 2013.
- [34] **Eclipse Foundation**, *Open Unified Process - Key Capabilities of the Unified Method Architecture (UMA)*, November 2008. http://epf.eclipse.org/wikis/openupsp/base_concepts/guidances/concepts/introduction_to_uma,_94_eo08LEdmKSqa_gSYthg.html, accessed November 2013.
- [35] <http://www.michael-richardson.com/>, *UMA versus RUP 2003 meta-model*, 2005. http://www.michael-richardson.com/rup_classic/core.base_concepts/guidances/concepts/uma_vs_rup_45521141.html, accessed November 2013.
- [36] **IBM Developer Works**, *IBM Rational Method Composer: Part 1: Key concepts*, December 2005. <http://www.ibm.com/developerworks/rational/library/dec05/haumer/index.html>, accessed November 2013.

- [37] **Sendall, S., & Kozaczynski, W.** *Model Transformation-the Heart and Soul of Model-Driven Software Development*. IEEE Software, 20(5), 42-45. 2003
- [38] **Vogel, O., Arnold, I., & Chughtai, A.** *Software Architecture: A Comprehensive Framework and Guide for Practitioners*. Springer. pp 258 -260, 2011.
- [39] **Pérez J. D.**, *Notaciones y lenguajes de procesos. Una visión global*. Department of Computer Languages and Systems of the University of Sevilla, 2006.
- [40] **Suryadevara, A.** *Surveying and Evaluating Tools for Managing Processes for Software-Intensive Systems* (Doctoral dissertation, Mälardalen University), 2010.
- [41] **World Wide Web Consortium XSL Transformations (XSLT)**, version 2.0. W3C Proposed Edited Recommendation, April 2009 <http://www.w3.org/TR/2009/PER-xslt20-20090421/>, accessed April 2013.
- [42] **World Wide Web Namespaces in XML 1.0**, August 2006. <http://www.w3.org/TR/2006/REC-xml-names-20060816>, accessed April 2013.
- [43] **World Wide Web XQuery 1.0 and XPath 2.0 Data Model (XDM)**, Second Edition, April 2009. <http://www.w3.org/TR/2009/PER-xpath-datamodel-20090421/>, accessed April 2013.
- [44] **Combemale, B., Crégut, X., Caplain, A., & Coulette, B.** *Towards a Rigorous Process Modeling with SPEM*. In ICEIS (3) (pp. 530-533). 2006.
- [45] **F. A. Zorzan and D. Riesco.** *Transformation in QVT of Software Development Process based on SPEM to Workows*. IEEE Latin America Transactions, 6(7), December 2008.
- [46] **M. Perez Cota, D. Riesco, I. Lee, N. Debnath, and G. Montejano.** *Transformations from SPEM Work Sequences to BPMN Sequence Flows for the Automation of Software Development Process*. J. Comp. Methods in Sci. and Eng., 10(1-2S1):61-72, September 2010.
- [47] **Portela, C., Vasconcelos, A., Silva, A., Sinimbú, A., Silva, E., Ronny, M., & Oliveira, S.** *A Comparative Analysis between BPMN and SPEM Modeling Standards in the Software Processes Context*. Journal of Software Engineering and Applications, 5(5), 330-339, 2012.

- [48] **White, S.** *Using BPMN to model a BPEL process.* *BPTrends*, 3(3), 1-18. 2005
- [49] **Jézéquel, J. M.** *A MDA Approach to Model & Implement Transformations.* Language Engineering for Model-Driven Software Development, 4101. Dagstuhl Seminar Proceedings **Dagstuhl, Germany**, 2004