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

TRANSFORMACIÓN AUTOMÁTICA DE MODELOS SPEM A BPMN CONSIDERANDO ROLES Y TAREAS COLABORATIVAS

TESIS PARA OPTAR AL GRADO DE MAGISTER EN CIENCIAS MENCIÓN COMPUTACIÓN

JUAN ANDRÉS PULGAR GONZÁLEZ

PROFESOR GUÍA:

MARÍA CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:

DANIEL PEROVICH GEROSA
ALCIDES QUISPE SANCA
MARCOS SEPÚLVEDA FERNANDEZ

SANTIAGO DE CHILE
2017

Resumen

La definición de un proceso de desarrollo de software aumenta las probabilidades de éxito de los proyectos, además de ser un medio que las empresas utilizan para obtener una posible certificación ISO o una evaluación CMMI. Para facilitar la definición del proceso se pueden utilizar diversas herramientas, entre las que destaca Eclipse Process Framework Composer, que se basa conceptualmente en el metamodelo SPEM/UMA.

Por otra parte, cuando se trata de definir procesos, especialmente procesos de negocio, es BPMN el lenguaje que se ha convertido en el estándar, proporcionando claridad gráfica y la posibilidad de ejecutar los procesos automáticamente sobre una plataforma BPMS. Sin embargo, BPMN no está definido específicamente para procesos de software y por lo tanto tiene menor especificidad que SPEM. Luego, si una empresa define su proceso de desarrollo de software utilizando el metamodelo SPEM no puede aprovechar directamente las bondades de BPMN.

Dada la problemática anterior, se plantea la posibilidad de realizar una transformación automática de un proceso de software modelado en SPEM/UMA a otro equivalente modelado en BPMN. La presente investigación cubre la realización de dicha transformación, junto con la respectiva validación del modelo resultante. Tomando en consideración que SPEM y BPMN tienen objetivos distintos, es probable que la transformación presente limitaciones relativas a que no existe una correspondencia entre todos sus elementos. También es parte de esta investigación identificar dichas limitaciones. Cabe mencionar que, si bien esta transformación ha sido abordada en otros trabajos, estos quedan en propuestas teóricas o bien sus resultados son parciales o no se encuentran disponibles.

Agradecimientos

En primer lugar a Dios que me ha permitido llegar hasta aquí.

A mi madre, Eliana y mi compañera de vida Carolina, quien me ha aguantado y apoyado durante todo este tiempo.

También agradezco a mi profesora guía, María Cecilia Bastarrica y al proyecto GEMS, quienes creyeron en mí y me dieron la oportunidad viajar a Río de Janeiro para participar en la conferencia “BPM 2016”.

Finalmente, a mis compañeros de trabajo, Rodrigo y Diego, quienes me apoyaron con puntos de vista diferentes y una que otra distracción.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

1 Introducción

1.1 Proceso de Desarrollo de Software

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 esperada del producto. La definición de un proceso de desarrollo de software aumenta la probabilidad de lograr el éxito en los proyectos de una empresa de desarrollo.

En 1989 se forma el consorcio Object Management Group (OMG), dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos. Es una organización sin fines de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones.

La recomendación del OMG para la definición de procesos de software es el metamodelo SPEM (Software Process Engineering Meta-Model). SPEM ha sido diseñado para describir procesos y sus componentes, siguiendo un enfoque de modelado orientado a objetos con base en UML, es decir que extiende los mecanismos de UML de una forma estandarizada, con el propósito de modelar procesos de desarrollo de software [1].

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, especificación y modelado de procesos de desarrollo de software:

- Eclipse Process Framework Composer[1]
- Objecteering Modeler[2]
- MagicDraw[3] UML (SPEM plug-in)
- Rational Method Composer[4]
- Enterprise Architect[5]

Entre los anteriormente nombrados destaca, principalmente por apegarse al estándar SPEM y ser de distribución gratuita, la herramienta Eclipse Process Framework Composer (EPF Composer) que integra componentes para la creación, configuración, visualización y publicación de métodos y procesos de desarrollo de software. Estas características, además del licenciamiento libre, han hecho que EPF Composer sea una herramienta de amplio uso en las PyMEs de software.

[1] http://www.eclipse.org/epf/
[2] http://www.objecteering.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/

1.2 Proceso de Negocio

En términos generales, un proceso es un conjunto de operaciones a las que se somete una entrada para elaborarla o transformarla. Desde la perspectiva de las empresas, un proceso de negocio es una colección de actividades que tomando una o varias clases de entradas crean una salida que tiene valor para un cliente [12].

Un proceso de negocio se puede modelar con diversos lenguajes y estándares tales como: Redes de Petri, Diagrama de Actividades de UML, BPMN (Business Process Modeling and Notation), XPDL (XML Process Definition Language), IDEF (Integration Definition) y EPC (Event-driven Process Chain), entre otros [33].

La recomendación de la OMG para la formalización de este tipo de procesos es BPMN, notación que en los últimos años se ha vuelto el estándar de facto debido principalmente a que [13]:

- Es una notación gráfica fácil de entender por analistas, implementadores y stakeholders.
- Se usa en una gran cantidad de organizaciones, estimándose, según algunas encuestas, que un 67% de las organizaciones utiliza BPMN como notación para la formalización de sus procesos [10].
- Reduce la distancia entre el diseño de un proceso de negocio y su implementación.

La facilidad de comprensión y su amplio uso, hacen de BPMN la notación ideal para comunicar y dar a conocer procesos de negocios al interior de una organización. Esta ventaja comunicacional de BPMN, puede conducir una mejora en la colaboración y coordinación de las distintas áreas de la organización, además de posibilitar la mejora continua de los procesos.

Adicionalmente, BPMN cuenta con el potencial de automatizar la ejecución de los procesos, mediante la implementación al interior de la organización de una plataforma BPMS. Un BPMS (Business Process Management Suite) es un sistema que permite coordinar la realización (ejecución) de procesos de negocio en base a representaciones de proceso explícitas (modelos). Existen numerosas herramientas BPMS para BPMN entre las que podemos destacar a Bizagi[6] y Bonita BPM[7].

El uso de un BPMS presenta diversos beneficios, entre los que se pueden mencionar [14]:

- Mediante un BPMS se consigue la automatización de los procesos de negocio. Esto tiene como consecuencia que los procesos están claramente definidos (y documentados), logrando una estandarización y homogeneización de los mismos.
- Con la ayuda de herramientas gráficas de definición del proceso se reduce el tiempo que hay que invertir en un cambio en el proceso derivado de un cambio en la lógica de negocio. Por lo tanto, se consigue una mayor agilidad y

[6] http://www.bizagi.com/es/
[7] http://es.bonitasoft.com/

flexibilidad para la organización, consiguiendo una ventaja competitiva respecto al resto de empresas.

- Por otra parte, es posible mediante los KPI (indicadores clave de rendimiento), la obtención de mediciones en tiempo real que permiten saber por ejemplo, cuáles son los cuellos de botella del proceso, cuánto se tarda en realizar ciertas tareas, etc.
- Asimismo, se pueden realizar simulaciones de la ejecución de los procesos ofreciendo información muy valiosa para optimizar el proceso incluso antes de ponerlo en producción, con el consiguiente ahorro en tiempo y dinero.

1.3 Proceso de Desarrollo de Software vs. Proceso de Negocio

Desde la perspectiva de la industria del software, los proyectos de desarrollo de software son instancias de procesos de negocio. Sin embargo, según la OMG, los procesos de desarrollo de software deberían ser modelados utilizando SPEM. Adicionalmente, 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 el cual debería estar definido en BPMN.

Dado lo anterior, utilizar BPMN para describir el proceso de desarrollo de software parece entonces una alternativa promisoria. Sin embargo, - ¿es posible utilizar BPMN para este propósito? Según [22], BPMN entrega los elementos de flujo necesarios para describir un proceso de desarrollo de software, sin embargo, tiene serias limitaciones con procesos altamente colaborativos.

Por otro lado, SPEM posee la especificidad suficiente para describir procesos de desarrollo de software, contando con los elementos de flujo necesarios y la capacidad de definir fácilmente tareas colaborativas, las cuales abundan en este tipo de procesos. Además la cercanía de SPEM a UML facilita su adopción por parte de las empresas desarrolladoras de software.

En este escenario, se hace necesario para la industria del software buscar una manera de aprovechar lo mejor de ambos mundos: la especificidad de SPEM y las ventajas comunicacionales y de automatización de BPMN.

1.4 Solución Propuesta

Con el objetivo de aprovechar las ventajas de BPMN, se propone crear un modelo en dicha notación tomando como punto de partida un modelo de proceso de desarrollo de software especificado en SPEM.

Para ello, se deben especificar reglas de transformación de los elementos SPEM a BPMN, junto con el flujo del proceso original. Un caso especial en esta transformación lo constituyen las tareas colaborativas, las cuales deben ser mapeadas de manera tal de no perder especificidad del modelo original especificado en SPEM.

Posteriormente, y con el objetivo de disminuir los errores potenciales que la aplicación manual de dichas reglas pueda generar, se procederá a construir una herramienta de software que permita realizar la transformación de forma automática.

Dicha herramienta de software trabajará con procesos de software formalizados mediante EPF Composer y deberá consumir los dos modelos generados por dicha herramienta (diagram.xmi y model.xmi) y mediante la aplicación de las reglas de transformación (en formato XSLT) deberá generar un modelo BPMN (Figura 1).

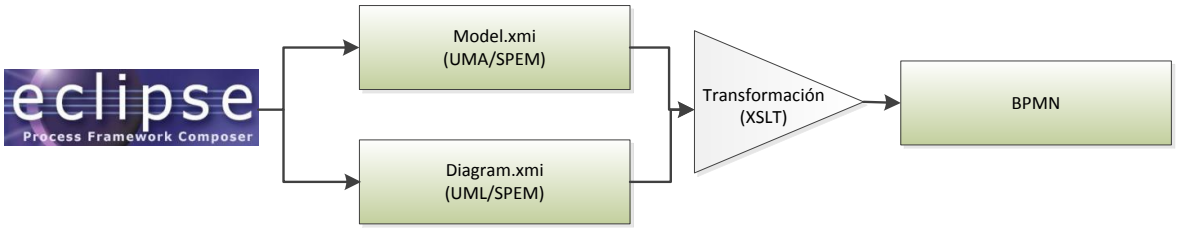


Figura 1: Esquema Solución Propuesta

Si se considera el objetivo del presente trabajo (facilitar la comunicación y potencial automatización de los procesos de desarrollo de software especificados en SPEM usando BPMN como estándar), la construcción de la transformación es un medio para determinar factibilidad, validar y detectar limitaciones en la transformación desde un modelo SPEM a BPMN, por lo que no se busca implementar una herramienta que sea de fácil mantención y evolución en el tiempo.

Dado lo anterior, para realizar la transformación entre modelos se utilizará una plantilla XSLT, en lugar de OCL o QVT (utilizados en trabajos relacionados), los cuales a pesar de tener un poder de expresividad mayor, resultan más complejos de utilizar. Además, XSLT entrega todo lo necesario, junto con las siguientes características:

- Simplicidad: Corresponde a un archivo de texto plano que puede ser trabajado con cualquier editor de texto, no necesitando componentes ni herramientas de software especiales.
- Flexibilidad: XSLT permite un amplio rango de resultados de transformaciones, no solo a archivos de tipos similares.
- Independencia de la plataforma: XSLT es independiente tanto del software como del hardware utilizado. Esto es una gran ventaja a la hora de adaptar la plantilla resultante a las necesidades de cada empresa.

Es importante mencionar que, si bien este tipo de transformación ha sido abordado en otros trabajos, estos quedan en propuestas teóricas o bien sus resultados son parciales o no se encuentran disponibles, lo que dificulta la aplicación por parte de las empresas de desarrollo de software. Adicionalmente, ninguno de los trabajos relacionados, disponibles en la literatura científica, aborda la transformación de tareas colaborativas.

Finalmente, es importante destacar que dado que SPEM y BPMN tienen objetivos distintos hay cierta información de SPEM que no tiene relación directa en BPMN, por lo

que probablemente no podrá ser llevado al modelo resultante. Asimismo, es posible que algunos elementos de BPMN deban ser agregados manualmente. Determinar estas y otras posibles limitaciones de la transformación, también es un objetivo de esta investigación.

1.5 Objetivos

1.5.1 Objetivo General

- Facilitar la comunicación y potencial automatización de los procesos de desarrollo de software especificados en SPEM usando BPMN como estándar.

1.5.2 Objetivos Específicos

- Definir reglas de transformación desde el metamodelo SPEM al metamodelo BPMN, manteniendo la información de roles incluso en tareas colaborativas.
- Construir una herramienta de software que permita realizar la transformación de forma automática utilizando las reglas de transformación definidas.
- Identificar las limitaciones de esta transformación.

1.6 Metodología a utilizar

El presente trabajo de investigación utiliza la metodología de casos de estudio descrita en [15]. Dicha metodología tiene las fases descritas en las secciones 1.6.1 a la 1.6.4.

1.6.1 Planteamiento

Con respecto a la pregunta de investigación, se debe considerar que el objetivo de la presente investigación es facilitar la comunicación y potencial automatización de procesos definidos en SPEM/UMA utilizando BPMN. Dado lo anterior, se hace necesario plantearse, *¿Es posible transformar un modelo SPEM/UMA a uno equivalente en BPMN?*

La pregunta anterior puede ser contestada utilizando la metodología de casos de uso, ya que la equivalencia de dos modelos de desarrollo de software está íntimamente relacionada a la organización que realiza la formalización de los procesos, por lo que para explicar dicho fenómeno el contexto es esencial y no puede ser controlado por el investigador (lo que descarta la simple experimentación).

1.6.2 Diseño

En esta fase se diseña el caso de uso y se busca asegurar la calidad de dicho diseño.

En esta investigación, la unidad a analizar es un proceso de desarrollo de software de una empresa chilena real formalizado en SPEM/UMA. Este proceso deberá ser transformado, mediante la aplicación de reglas, a un proceso formalizado en BPMN. En este contexto aparece la transformación propuesta en la sección 1.4. Es importante destacar que, para efectos de esta investigación, dicha transformación representa un medio y no un fin.

Con respecto a la calidad del diseño de los casos de uso propuestos, [15] propone algunos criterios:

1. Validez de Construcción: Este criterio permite identificar las medidas operativas correctas para los conceptos que se están estudiando. Con respecto a la presente investigación, este criterio se cumple debido a que establece una "cadena de evidencia", donde se puede tomar el proceso especificado en EPF Composer directamente, transformarlo (de forma automática con la propuesta de la sección 1.4) y obtener la transformación BPMN. De esta manera no existe la posibilidad de manipulación de la entrada al proceso.

2. Validez Interna: Este criterio busca establecer relaciones causales, donde se cree que ciertas condiciones llevan a otras. En el caso de esta investigación, este criterio no aplica.
3. Validez Externa: Este criterio define el dominio en el que los resultados de un estudio pueden ser generalizados. Para cumplir con este criterio, se toman al menos dos casos de estudio (ver sección 4.1), de esta manera, se puede usar la "lógica de replicación".
4. Confiabilidad: El objetivo de este criterio es estar seguro de que si otro investigador toma el mismo caso de estudio, llegue a las mismas conclusiones. Aquí la propuesta automática de transformación (sección 1.4) cumple, una vez más, un papel fundamental, al permitir obtener la misma transformación cada vez.

1.6.3 Preparación

En esta fase, se prepara la recolección de los casos de estudio. Esto tanto en términos del investigador, como de los casos de estudio en sí.

Para ampliar los conocimientos relacionados con la investigación y preparar al investigador, se realiza un estudio de los temas relacionados (SPEM/UMA, BPMN, XSLT, etc.) que se encuentra disponible en el Capítulo 2.

Posteriormente, se realiza la transformación automática de SPEM/UMA a BPMN (Capítulo 3). Esta transformación será parte integral del protocolo de casos de estudio que se puede encontrar en la sección 3.2.2. La transformación fue probada con diversos casos de estudio piloto generados en el laboratorio.

Finalmente, se busca la participación de participantes claves al interior de la organización desde donde se obtendrán los casos de uso, con lo que se consigue la participación de Jefaturas y Gerencias.

1.6.4 Recolección, Análisis y Publicación

En la fase de Recolección se obtienen los casos de estudio. Una completa descripción se encuentra en las secciones 4.1 y 4.2.

Posteriormente, en la fase de Análisis se analizan los resultados obtenidos (véase la sección 4.3). Finalmente, en la fase de Publicación se publican las conclusiones de la investigación (disponible en el Capítulo 5).

2 Marco Teórico

2.1 Software Process Engineering Metamodel (SPEM)

Según el OMG [1], SPEM es un meta modelo de ingeniería de procesos y un marco de trabajo conceptual que provee los conceptos necesarios para modelar, documentar, presentar, administrar, intercambiar e instanciar métodos y procesos de desarrollo de software. Una implementación de este meta modelo debería ser entregada a los ingenieros de proceso, líderes y gerentes de proyecto responsables de mantener e implementar procesos para sus organizaciones o proyectos individuales.

SPEM fue creado por el OMG como un estándar de alto nivel para describir procesos utilizados en el desarrollo de software. Inicialmente fue creado como un metamodelo independiente pero más tarde fue reformulado con la estructura de un perfil (profile) UML. Esto significa que se ajustaron los conceptos orientados a procesos para representar conceptos orientados al modelado [2].

En la base de SPEM está la idea de que un proceso de desarrollo de software es una colaboración entre las entidades activas abstractas llamadas roles del proceso que realizan operaciones llamadas actividades en entidades concretas, tangibles llamadas productos de trabajo. Múltiples roles interactúan o colaboran para intercambiar productos de trabajo y provocar la ejecución, o representación de ciertas actividades. El objetivo principal de un proceso es traer unos productos de trabajo a un estado bien definido. Para esto, un primer paso consistirá en reestructurar el rol, la actividad y el producto de trabajo. Las actividades utilizan productos de trabajo con el propósito de crear nuevos productos de trabajo. El trabajo es realizado usualmente por personas que desempeñan roles. Ellos realizan las actividades y son responsables de mantener directamente el conjunto de productos de trabajo.

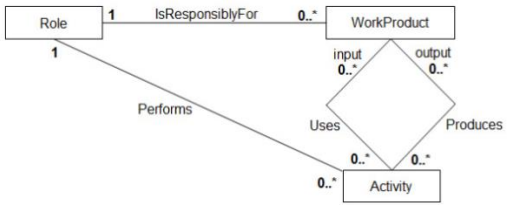


Figura 2: Relación de elementos fundamentales de SPEM [2]

La mayor unidad de trabajo es una actividad, la cual puede estar descompuesta en más sub actividades y tareas. Las clases que se aprecian en la Figura 2 describen el trabajo que se necesita realizar. El realizador de esta unidad de trabajo es el ProcessRole, el cual es un tipo de realizador de procesos, los cuales consisten en un conjunto de WorkDefinition (Figura 3). Las personas interpretan roles para crear WorkProduct los cuales están asociados a WorkDefinition (incluyendo tareas y a actividades).

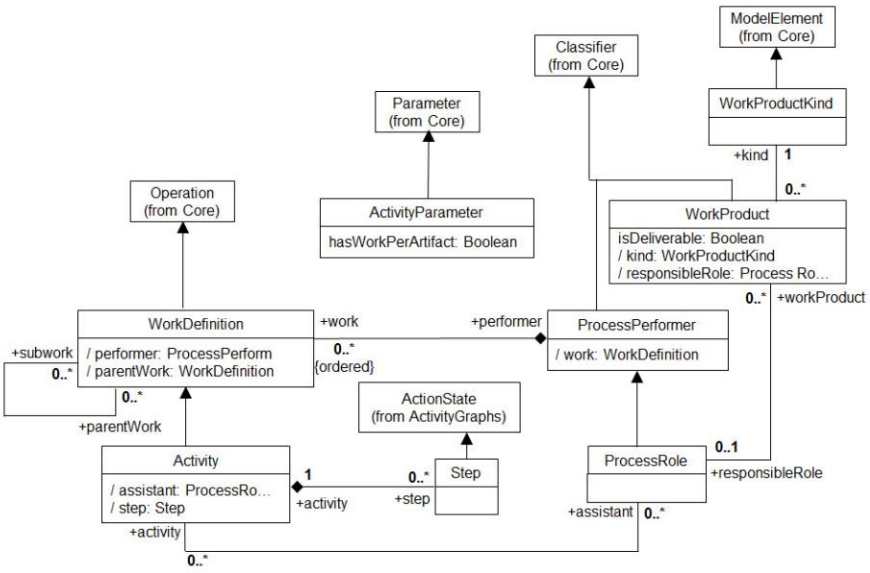


Figura 3: Taxonomía de Actividad en SPEM [2]

De esta manera, en SPEM, un proceso se puede ver como una colaboración entre roles para alcanzar cierta meta u objetivo. Para dirigir su representación, podemos restringir la orden en la cual las actividades deben ser, o pueden ser ejecutadas. También es necesario definir la "forma" del proceso en un determinado tiempo, y la estructura del ciclo de vida en términos de fases y de iteraciones (Figura 4).

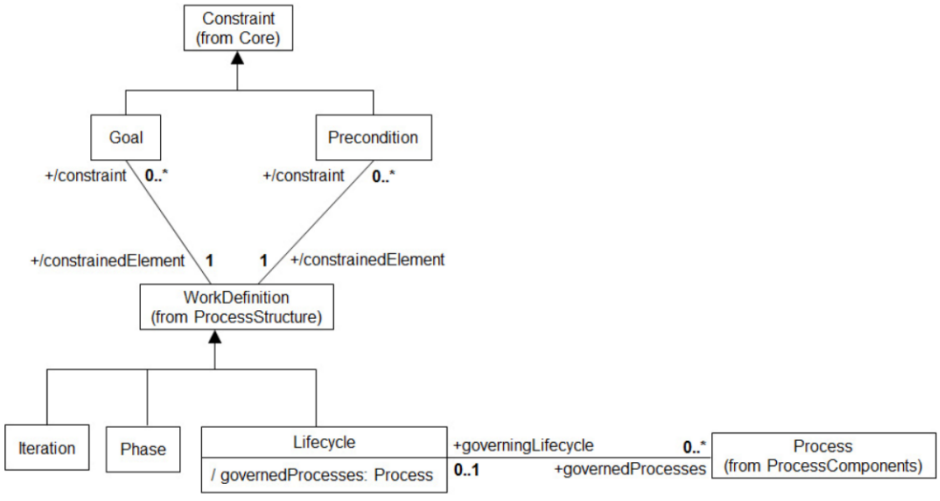


Figura 4: Taxonomía de Ciclo de Vida en SPEM [2]

2.1.1 SPEM v1.1

En la Tabla 1 se pueden apreciar los estereotipos gráficos utilizados por SPEM v1.1 para construir procesos de desarrollo de software [1]. A pesar de ser liberada desde

2002 y tener una revisión en el 2005 (SPEM 1.1), son pocas las implementaciones de SPEM 1.x en herramientas comerciales, y las que existen lo soportan de forma parcial. La causa de esta situación parece ser lo difícil que resultó su implementación, además de que algunos elementos de la semántica son ambiguos. En la Figura 5 se pueden apreciar dos ejemplos de diagramas construidos en SPEM v1.1 [2].

Elemento	Símbolo
Activity	
Document	
Guide	
Process	
Phase	
Process Package	
Rol	
WorkProduct	
WorkDefinition	
Model	

Tabla 1: Listado de estereotipos de SPEM v1.1

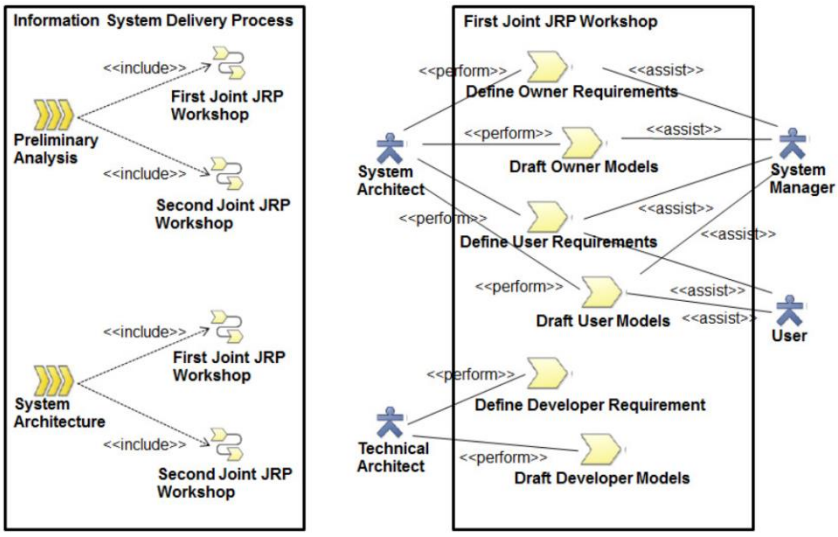


Figura 5: Diagrama de Casos de Uso en SPEM v1.1 [2]

2.1.2 Unified Method Architecture (SPEM/UMA)

UMA es una evolución de estándar SPEM v1.1 [4] y ha sido desarrollado como unificación de diferentes métodos y lenguajes de ingeniería de proceso tales como [3]: la extensión SPEM para UML, los lenguajes usados para IBM Rational RUP v2003, el proceso unificado, y el método de Servicios Globales de IBM.

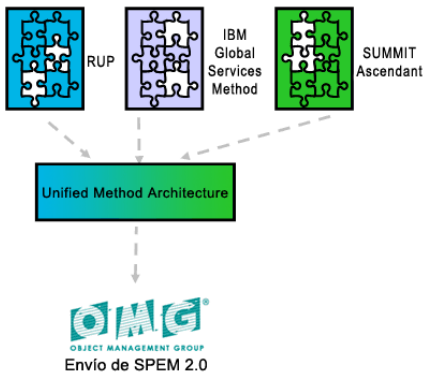


Figura 6: Modelos Fuente para UMA [5]

En 2005, IBM y otros asociados de la OMG comenzaron a trabajar para convertir UMA, junto a algunas mejoras sugeridas por los asociados, en SPEM v2.0, la cual es liberada en Abril de 2008 [1].

Las principales características de UMA son las siguientes:

- **Separación del Contenido de método y Proceso**: UMA proporciona una clara separación de las definiciones del contenido del método (Method Content) de su aplicación en los procesos. Esto se obtiene a través de la definición por separado de una:

    - Base de conocimiento de desarrollo reutilizable (Method Content) , en forma de descripciones de contenido general como roles, tareas, productos de trabajo y guías
    - Aplicación específica para cada proyecto donde se combinen y reutilicen los elementos del Method Content

- **Reutilización del contenido**: UMA permite que cada proceso haga referencia a contenido de método en un repositorio 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 tales como añadir un producto de trabajo a una tarea, renombrar a un rol, o eliminar pasos de una tarea.

- **Familias de 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. Además, permite obtener distintas variantes de procesos específicos construidos mediante la reutilización dinámica del mismo Method Content y los mismos Capability Patterns, pero aplicada con distintos niveles de detalle y de escala a través de algunos mecanismos de variabilidad.

- **Múltiples ciclos de vida**: 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.

- **Extensibilidad y mecanismos de plug-in**: UMA proporciona dos tipos de plug-ins que trabajan de modos distintos. 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 en responsabilidades, define los roles necesarios y los productos de trabajo asociados.

- **Patrones de Procesos reutilizables**: Los Capability Patterns son bloques de construcción reusables para crear nuevos procesos de desarrollo. Seleccionar y aplicar uno puede ser realizado mediante:

    - La acción de copiar y modificar, lo que permite al ingeniero de procesos personalizar el contenido acorde a sus necesidades de aplicación.
    - Vinculación dinámica, lo que permite aplicar muchas veces el mismo patrón en un proceso. Si el patrón se modifica, dicha modificación se verá reflejada en todas las aplicaciones del patrón en el proceso.

2.1.3 Descripción de comportamiento en SPEM/UMA

SPEM no pretende ser un lenguaje genérico de modelado de procesos, por lo que no proporciona un modelo de comportamiento. En vez de eso, permite a los modeladores seleccionar su propio modelo de comportamiento (por ejemplo UML o BPMN).

UMA utiliza la especificación de comportamiento Activity de UML para la definición del proceso.

Un Activity es un comportamiento especificado como secuencia de unidades subordinadas, utilizando un modelo de flujo de datos y control. Los comportamientos subordinados coordinados por estos modelos pueden iniciarse porque otros comportamientos en el modelo terminan de ejecutarse, porque los objetos y los datos están disponibles o porque los eventos ocurren externamente al flujo.

El flujo de ejecución se modela como ActivityNodes conectados por ActivityEdges. Los nodos Activity también incluyen construcciones de flujo de control, como sincronización, decisión y control de concurrencia (Figura 7).

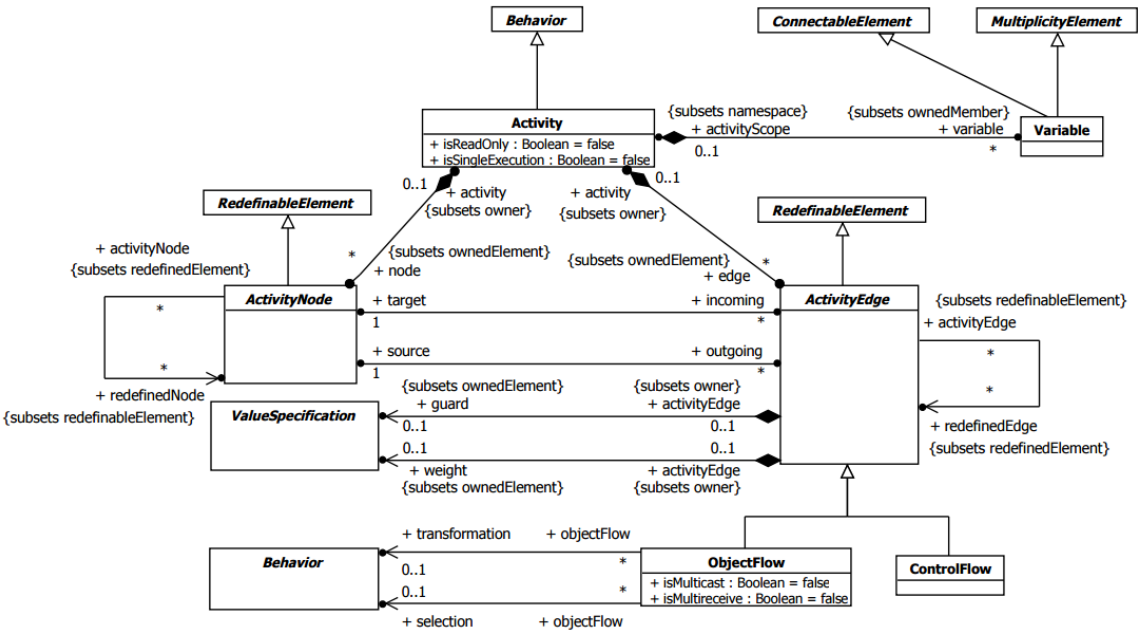


Figura 7: Sintaxis abstractas de Activity

Un ControlNode es un tipo de ActivityNode que se utiliza para gestionar el flujo entre otros nodos en un Activity. Entre los diversos nodos de control, se incluyen los InitialNodes, FinalNodes, ForkNodes, JoinNodes, MergeNodes y DecisionNodes (Figura 8).

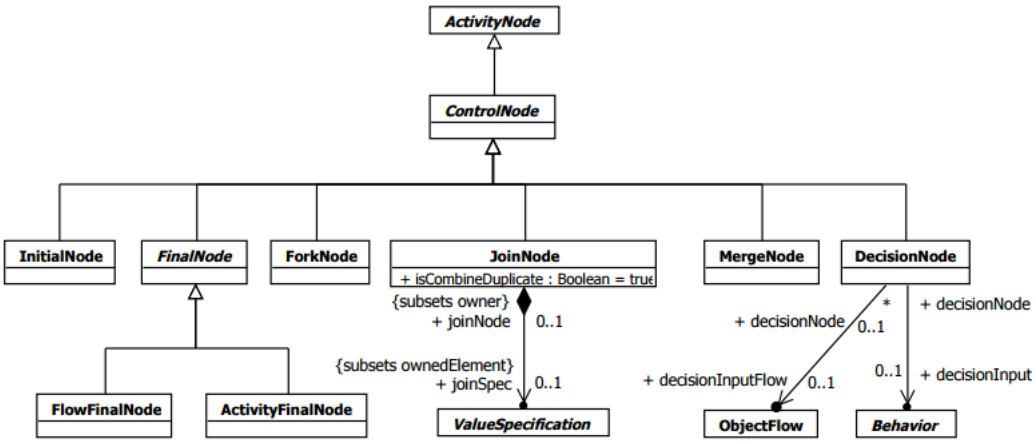


Figura 8: Sintaxis abstracta de ControlNode

2.1.4 SPEM v2.0

Además de las ventajas aportadas por la inclusión de UMA a SPEM (detalladas en la sección anterior), se agregan otras características y mejoras a SPEM 2.0 con respecto a sus versiones previas, tales como [1]:

- Al basarse en UML 2.0 toma ventaja de su nueva funcionalidad en la mejora de las técnicas y capacidades del modelado de procesos.
- Define un nuevo esquema XML SPEM, basado en MOF 2.0. Los esquemas XML proporcionan una mayor riqueza y control que en SPEM 1.1.
- Proporciona una guía para la migración de modelos de procesos de SPEM 1.1 a SPEM 2.0.
- Considera la retroalimentación de las primeras implementaciones para identificar inconsistencias relacionadas con la funcionalidad y practicidad de SPEM 1.1.
- Define extensiones para que SPEM utilice herramientas de automatización de procesos.
- Establece una relación más estrecha con otros estándares adicionales a UML como Business Process Definition Meta-model y Business Process Runtime Interfaces.
- Define extensiones al metamodelo de procesos que pueden ser usados tanto en los procesos de desarrollo de software como en los procesos de ingeniería de sistemas.

SPEM 2.0 está estructurado en siete paquetes de metamodelos principales. La estructura divide el modelo en unidades lógicas. Cada unidad extiende la unidad de la que depende, proporcionando estructuras y capacidades adicionales a los elementos definidos:

1. Core: contiene las clases y abstracciones del metamodelo que constituyen la base para crear todas las clases de los demás paquetes.

2. Process Structure: Establece la base para todos los modelos de proceso y soporta la creación de estos en una forma fácil y flexible.
3. Process Behavior: Representa un proceso como una estructura estática, permitiendo anidar actividades y definir dependencias entre ellos.
4. Managed Content: Establece conceptos para la gestión de descripciones y contenido textual que apoyen a los conceptos de un proceso.
5. Method Content: Permite la construcción de una base de conocimientos del desarrollo definiendo conceptos del ciclo de vida y métodos, técnicas y las mejores prácticas.
6. Process with Methods: Define estructuras para integrar procesos definidos con Process Structure con conceptos de Method Content.
7. Method Plugin: Introduce conceptos para diseñar y gestionar repositorios de procesos reutilizables

En la

Tabla 2 se aprecian los símbolos más utilizados del estereotipo UML de SPEM v.2.0. Para mantener compatibilidad con SPEM v1.1, la mayoría de los estereotipos tienen su contraparte, sin embargo, dado que SPEM v2.0 introduce nuevos conceptos, no todos tienen su equivalente en SPEM v1.1, tal como se aprecia en la

Tabla 3 [2].

Elemento	SPEM v2.0
Activity	
Category	
CompositeRole	
Guidance	
Method Content Package	
MethodLibrary	
MethodPlugin	
Milestone	
Process	
ProcessComponent	
ProcessPackage	
Role Definition	
RoleUse	
Step	

TaskDefinition	
TaskUse	
WorkProductDefinition	
WorkProductUse	

Tabla 2: Estereotipos UML de SPEM v2.0

Elemento	SPEM v1.1	SPEM v2.0
Activity		
Category		
Guidance		
Process Component		
Role		
Step		

Tabla 3: Comparación entre SPEM v1.1 y SPEM v.2.0

En la Figura 9 se puede apreciar un ejemplo de diagrama construido utilizando el estereotipo UML de SPEM v2.0.

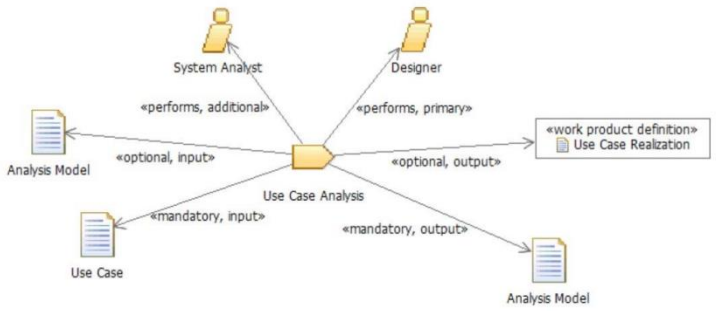


Figura 9: Ejemplo de uso de SPEM v2.0

2.2 Eclipse Process Framework Composer

Eclipse Process Framework Composer (EPF Composer) es una herramienta para ingenieros de procesos, líderes y administradores de proyectos, quienes son responsables de mantener e implementar procesos para organizaciones dedicadas al

desarrollo o para proyectos individuales. Se distribuye de forma gratuita bajo licencia GNU.

Eclipse EPF Composer tiene dos propósitos básicos [6]:

- Proveer a los desarrolladores con una base de conocimiento de capital intelectual que les permita buscar, administrar y desplegar contenido. Esta base de conocimiento puede ser usada como referencia y material educativo, y forma la base para el proceso de desarrollo. EPF Composer está diseñado para ser un administrador de contenido que provee una estructura de administración y un aspecto comunes para todo el contenido, en vez de ser un sistema administrador de documentos en el cual se almacenan y se acceden documentos difíciles de mantener, que tienen cada uno su propia forma y formato.
- Proveer capacidades de ingeniería de procesos para la selección, adecuación y rápido ensamblado de procesos para proyectos de desarrollo concretos. EPF Composer tiene catálogos de procesos predefinidos para situaciones típicas de procesos que pueden ser adaptados a necesidades individuales.

Entre los principales objetivos del EPF Composer están un conjunto de necesidades comunes de los equipos de desarrollo cuando se enfrentan a la asimilación de métodos y procesos entre las que se pueden citar:

- Acceso fácil y centralizado a la información.
- Formatos estándar que permitan una fácil integración.
- Base de conocimiento actualizada para que ellos mismos aprendan sobre métodos y mejores prácticas.
- Soporte para dimensionar correctamente sus procesos.
- Habilidad de estandarizar prácticas y procesos dentro de las organizaciones.
- Cerrar la brecha entre la ingeniería de procesos y el establecimiento de los mismos en las organizaciones por medio del uso de representaciones y terminologías similares.

EPF Composer se alinea con el metamodelo SPEM v2.0, en su versión obtenida desde UMA, por lo que describe un proceso esencialmente a través de la creación de un MethodContent y los respectivos Process (Figura 10).

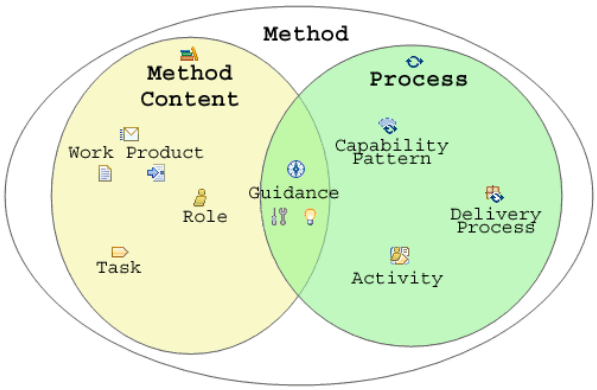


Figura 10: Vista general de trabajo de EPF Composer

Dado lo anterior, para crear un modelo de desarrollo de software en EPF Composer, se debe crear una Librería de Método, dentro de la cual se debe crear el o los plug-in de procesos correspondientes y la configuración. Dentro del plug-in de procesos, se debe crear el Method Content (con sus respectivos elementos) y el Process correspondiente que utilice los elementos definidos en el Method Content (Figura 11).

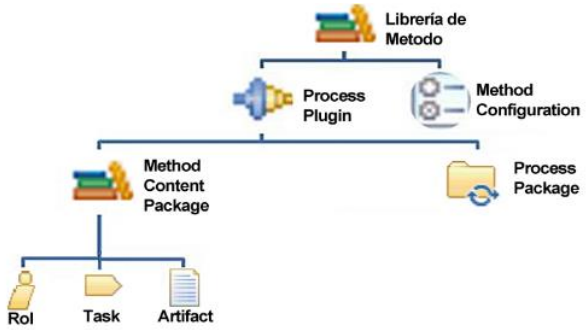


Figura 11: Esquema de creación de un modelo en EPF Composer

2.2.1 Arquitectura de EPF Composer

EPF Composer es una aplicación Java construida sobre una colección de componentes y estándares libres. La Figura 12 muestra la organización de los componentes claves en su arquitectura.

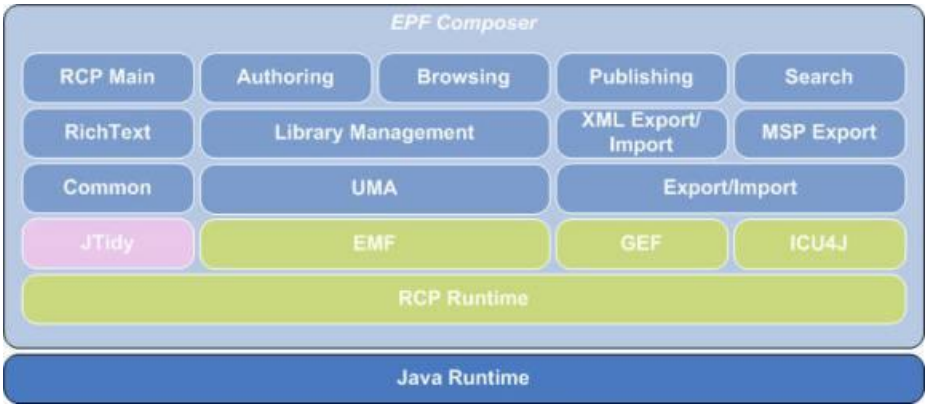


Figura 12: Arquitectura de EPF Composer

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 [7]:

- Eclipse Rich Client Platform (RCP): Plataforma extensible para construir aplicaciones Java 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 Unicote 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 impresión 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-in y una nueva Method Configuration.
    - 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. 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.
- 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.
- 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.

2.2.2 Estructura de archivos de EPF Composer

EPF Composer crea un directorio por cada proyecto, donde se pueden encontrar los archivos que lo componen. Entre los archivos se destacan [8]:

- **library.xmi**: Este archivo contiene una referencia a cada plug-in del Method Library. Cuando se crean nuevos procesos o paquetes EPF Composer no modifica este archivo. Este archivo es compartido por todos los plug-in del Method Library.
- **plug-in.xmi**: Este archivo contiene una referencia a cada elemento del plug-in, así como nombres de presentación, descripciones breves, relaciones entre elementos, entradas y salidas, responsables, etc. Hay un archivo plugin.xmi por cada plug-in.
- **model.xmi**: Este archivo contiene una referencia a los descriptores (roles, tareas y productos de trabajo) además de nombres y descripciones breves de cada uno. Hay un archivo model.xmi por cada proceso (capability pattern o delivery process).
- **content.xmi**: Este archivo contiene el texto descriptivo de los elementos de proceso, con la excepción de los nombres, nombres de presentación y descripciones breves. Hay un archivo content.xmi por cada proceso (capability pattern o delivery process).
- **diagram.xmi**: Todos los diagramas se guardan en este archivo.

2.3 BPMN

La primera versión de BPMN (Business Process Modeling Notation) fue desarrollada por el instituto BPMI (Business Process Management Initiative) principalmente bajo la tutela de Stephan A. White, profesional de la IBM en 2004. Desde un principio su

principal objetivo fue ofrecer una notación gráfica, estandarizada que permitiera automatizar los procesos de negocio a partir del diseño gráfico. En el año 2005 fue trasladado el proyecto a la OMG (Object Management Group), debido a que BPMI no era un instituto de administración de estándares. A través de la OMG, BPMN se difundió rápidamente a nivel mundial y casi todos los proveedores, ya sean grandes o pequeños, académicos o consultores empezaron a adoptar este estándar [9].

La Figura 13 muestra la evolución de BPMN hasta la versión 2.0 usada actualmente. Paradojalmente, hasta la versión 1.2 no se podían mapear los modelos directamente en un entorno técnico. Debido a esta falencia existieron muchos problemas en convertir los modelos a lenguajes de ejecución como BPEL. Recién con la versión 2.0 existe un metamodelo que permite ejecutar directamente los modelos creados en BPMN, esto unido a la estandarización conlleva los siguientes beneficios [9]:

- Aumenta el grado de independencia de las herramientas BPM. Al 2010 existen más de 60 herramientas de modelación BPMN y muchas de ellas gratuitas (como Bonita BPM).
- La comunicación entre los stakeholders de un proyecto será más rápida, fluida y expresiva.
- Se puede esperar un crecimiento en el nuevo personal con conocimientos en BPMN.
- Institutos, universidades, y empresas consultoras van a invertir recursos en formar profesionales con conocimientos en esta notación.

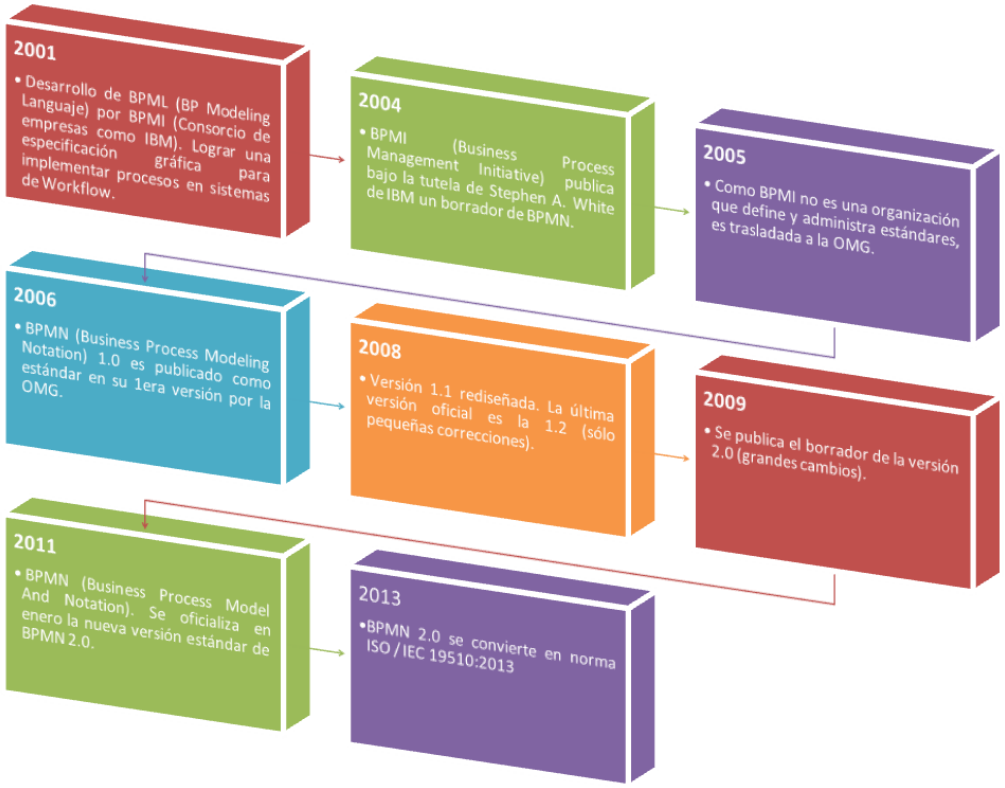


Figura 13: Evolución de BPMN [11]

2.3.1 Marco Estructural

En [10], se propone un marco estructural para el trabajo con BPMN. Este marco permite seleccionar el tipo de objetos y patrones que se deben utilizar o no es recomendable utilizar de acuerdo al nivel [11]:

- Nivel 1 (Modelo Macroproceso): La audiencia del nivel descriptivo son principalmente los Procesos Owner y Process Manager como también Process Participants y Process Analysts en las primeras fases de un proyecto. En este nivel queremos definir el contexto de los procesos que se deben levantar, modelar, documentar y eventualmente rediseñar. El objetivo de este nivel es además validar el alcance y la funcionalidad principal de los procesos que deben levantarse. En este nivel describimos el flujo normal del proceso, así como queremos que ocurra, sin considerar casos de excepción o errores. El nivel descriptivo nos sirve para validar en forma rápida el alcance del proyecto con los responsables del negocio e introducir al resto de los participantes en él.
- Nivel 2 (Modelo Operativo): En el nivel operacional se desarrolla toda la lógica de los procesos en su máximo detalle, incluyendo los casos de excepción, fallas e interrupciones que pueden ocurrir en nivel de negocio. La habilidad del analista de procesos consiste en desarrollar un modelo en el nivel 2 que abarque toda la lógica de negocio y que sea transferible al siguiente nivel de implementación.
- Nivel 3a (Modelo Técnico): El modelo técnico es la representación del modelo operacional en un Process Engine, pero adaptando el proceso de negocio a un modelo ejecutable y enriqueciéndolo con aspectos técnicos. Como en realidad no siempre se implementan los modelos de negocio con un Process Engine, dividimos este nivel en un modelo técnico (3a), en el cual en un principio se sigue detallando el modelo de negocio en un BPMS y la opción de un desarrollo propio (3b). En BPMN podemos especificar el modelo técnico directamente en un Process Engine a partir de BPMN 2.0.
- Nivel 3b (Especificación Técnica): Si no se utiliza un Process Engine, la lógica de negocio tiene que ser desarrollada en algún lenguaje de programación. En estos casos hay que elaborar una especificación técnica, que no tiene mucha relación con BPMN. Los diagramas deben pasarse a una especificación adecuada para el ambiente de programación escogida.
- Nivel 4a (Programación): Luego de la especificación del nivel 3b es necesario implementar técnicamente el proceso en una plataforma “tradicional”. Si se utiliza un Process Engine no es necesario elaborar una especificación para el desarrollo, razón por la cual el esquema de la pirámide se muestra en forma asimétrica.

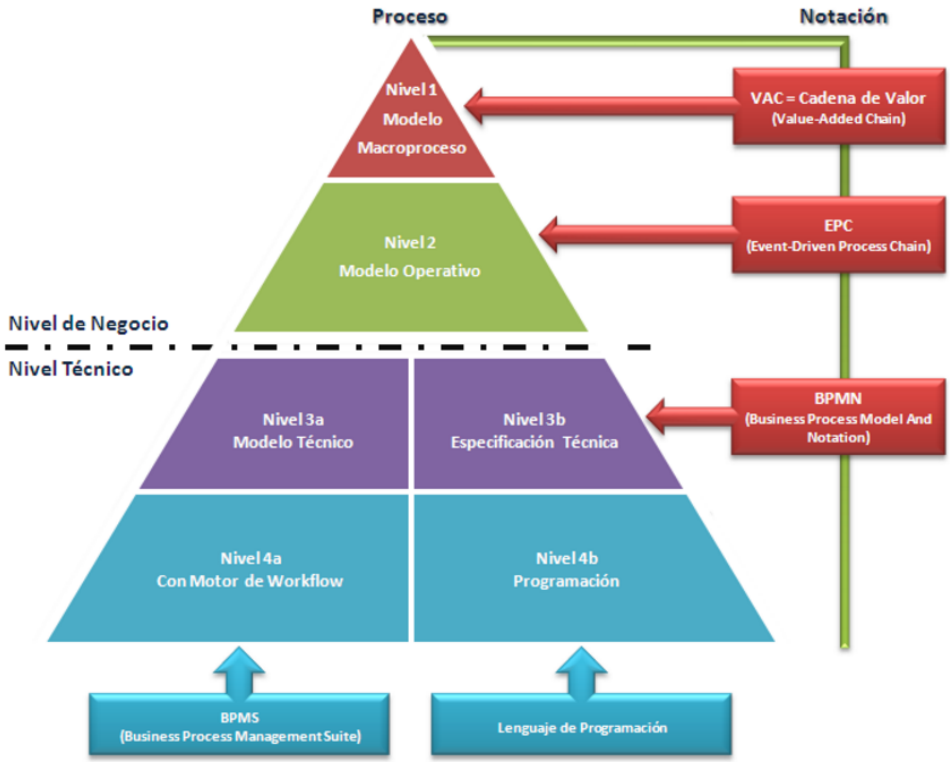


Figura 14: Marco Estructural de BPMN [11]

2.3.2 Diagramas BPMN

Los elementos gráficos en BPMN se encuentran clasificados dentro de 4 categorías [10][11]:

- Objetos de Flujo: Son los principales elementos gráficos que definen el comportamiento de los procesos. Pueden ser clasificados en eventos, actividades y compuertas. La Tabla 4 muestra los principales objetos de flujo utilizados en BPMN.

Tipo	Características	Íconos
Eventos	- Representan algo que ocurre o que puede ocurrir durante el curso de un proceso. - Pueden ser de tres tipos: inicio, intermedio y finalización	
Actividades	- Representan el trabajo realizado dentro de una organización. - Consumen recursos. - Pueden ser simples o compuestas.	

Compuertas	- Son los elementos para controlar puntos de divergencia/convergencia del flujo.	

Tabla 4: Principales Objetos de Flujo BPMN

- Objetos de Conexión: Son los elementos gráficos usados para conectar dos objetos del flujo de un proceso. Pueden ser clasificados en Líneas de Secuencia, Líneas de Mensaje, Asociación de Datos o Asociaciones. La Tabla 5 muestra los objetos de conexión utilizados BPMN.

Tipo	Descripción	Ícono
Líneas de Secuencia	- Representan el control de flujo y la secuencia de las actividades. - Se utiliza para representar la secuencia de los objetos de flujo, donde encontramos las actividades, las compuertas y los eventos.	
Líneas de Mensaje	- Las líneas de mensaje representan la interacción entre varios procesos o pools. - Representan Señales o Mensajes NO flujos de control. - No todas las líneas de mensaje se cumplen para cada instancia del proceso y tampoco se especifica un orden para los mensajes.	
Asociaciones	- Se usan para asociar información adicional sobre el proceso. - También se usan para asociar tareas de compensación.	

Tabla 5: Objetos de Conexión BPMN

- Canales: Son los elementos gráficos utilizados para organizar las actividades del flujo en diferentes categorías visuales que representan áreas funcionales, roles o responsabilidades. La Tabla 6 muestra los canales utilizados en BPMN.

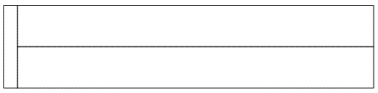
Tipo	Descripción	Ícono
Pool	- Actúa como contenedor de un proceso. - El nombre del pool puede ser el del proceso o el del participante. - Siempre existe al menos uno, así no se diagrame. - Figurativamente representa una Piscina.	
Lane	- Subdivisiones del Pool. - Representan los diferentes participantes al interior de una organización. - Figurativamente representa el Carril de la Piscina.	

Tabla 6: Canales en BPMN

- Artefactos: Son elementos gráficos utilizados para proveer información adicional sobre el proceso. Pueden ser de 3 tipos: Objetos de Datos, Grupos o Anotaciones (Tabla 7).

Tipo	Descripción	Ícono
Objetos de Datos	- Representan los documentos, la información y otros objetos que son usados o actualizados como durante el proceso. - Los objetos de datos no afectan directamente los flujos de secuencia o los flujos de mensajes del proceso.	

Grupos	- Se utiliza para agrupar un conjunto de actividades, ya sea para efectos de documentación o análisis, no afecta la secuencia del flujo.	
Anotaciones	- Son utilizados para proporcionar información adicional sobre el proceso.	

Tabla 7: Artefactos en BPMN

2.3.3 Modelado de actividades colaborativas en BPMN

En BPMN, un pool es un elemento que define los límites de un proceso de negocio. Un pool contiene al menos un proceso de negocios, mientras que un lane (carril) es una subdivisión dentro de un pool que se utiliza para organizar y categorizar actividades de un proceso, sin embargo, es difícil pensar un carril sólo como eso, ya que al momento de modelar se tiende a asignarles un nombre y significado, generalmente asociado a roles.

La práctica anterior se vuelve importante, si se considera que el 90% de los modeladores utilizan BPMN como un medio gráfico para documentar procesos, mientras que sólo el 10% restante está realmente interesado en simular, ejecutar y exportar procesos de negocios usando BPMN [19]. Incluso las prácticas comunes indican a los modeladores que asignen roles a los carriles, utilizándolos como más que simples contenedores.

Considerando lo anterior, es deseable que el presente trabajo cubra la práctica más reconocida a la hora de aplicar roles sobre las actividades, utilizando los carriles para representar roles. Sin embargo, esto presenta un problema: ¿Cómo modelar una tarea colaborativa si una tarea sólo puede estar en un carril? En [19] se exploran las 4 alternativas más aceptadas para resolver este problema, las cuales son presentadas a continuación.

2.3.3.1 Duplicar tareas en cada carril

Consiste en duplicar la tarea en cada carril involucrado agrupándolas, de ser necesario, para mejorar la lectura del diagrama (Figura 15).

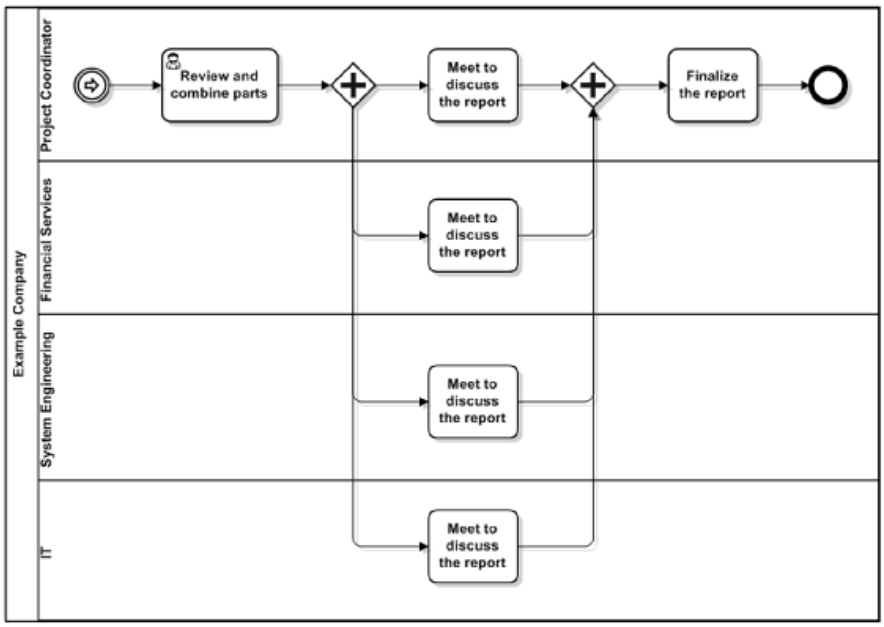


Figura 15: Ejemplo de duplicación de actividades [19]

Esta solución tiene varios problemas. Las tareas duplicadas no son copias de un original (aunque conceptualmente lo parezca), resultando en un diagrama con múltiples tareas ejecutando la misma acción en paralelo. Esta solución obliga a los modeladores a poner cuidado en la coherencia, garantizando que todas las propiedades sean las mismas para todas, poniendo una dificultad extra a la hora de realizar modificaciones. Incluso, desde el punto de vista gráfico, esta solución presenta problemas haciendo que tareas que no estén en carriles contiguos, sean de difícil seguimiento. Además, será necesario sincronizar todas las actividades con una compuerta paralela.

2.3.3.2 Asignar diferentes tareas a los roles

Para resolver los problemas de la primera solución, y suponiendo que se pueden realizar pequeños ajustes a la descripción del modelo, se podría asignar diferentes tareas a cada rol, dividiendo la tarea original en tareas más pequeñas (Figura 16).

Esta solución tiene menos problemas que la anterior, aunque aparecen nuevas dificultades. El modelador debe “inventar” nuevas tareas, aunque estas no tengan su reflejo en el mundo real (en el caso de una reunión, varios roles pueden estar solo escuchando, teniendo el modelador que crear actividades diferentes para cada uno). Por otro lado, al mirar el diagrama, la primera impresión no es la de una reunión, sino de diferentes actividades pasando al mismo tiempo. Finalmente esta alternativa no es fácilmente automatizable, ya que la división de tareas debe ser hecha por un ser humano.

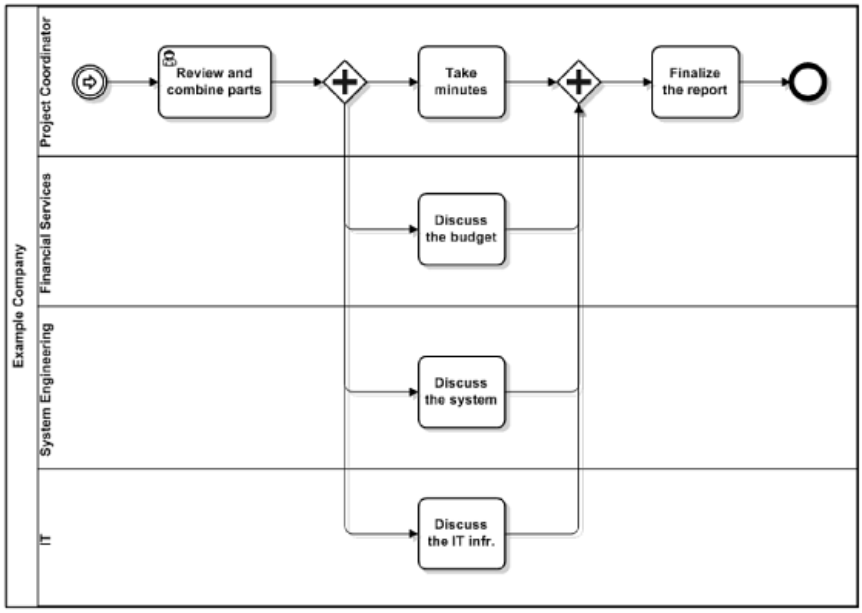


Figura 16: Ejemplo de asignación de roles [19]

2.3.3.3 Crear un carril para representar un grupo de roles

Consiste en crear un nuevo carril dentro del proceso para representar un grupo de roles involucrados en una tarea colaborativa. Esto permite evitar la repetición y reducir el número de tareas (Figura 17).

Sin embargo, esta solución también presenta inconvenientes. La complejidad del modelo se incrementa debido a la adición de un nuevo carril, que quizás sólo contendrá una sola tarea. Por otro lado, los modeladores pueden tener problemas si existen diferentes grupos de roles, ya que cada vez que una actividad sea realizada por un grupo diferente, un nuevo carril debe ser agregado al modelo.

2.3.3.4 Crear un nuevo proceso para cada actividad colaborativa

Consiste en utilizar un llamado a un proceso diferente donde la actividad colaborativa ha sido modelada en un pool diferente que contiene un solo carril representando al grupo de roles que llevará a cabo la tarea. Es importante destacar que la llamada al proceso está contenida en el carril del rol responsable de liderar la reunión. En otras palabras, los modeladores pueden crear tareas colaborativas en un proceso diferente (uno por cada tarea y grupo de roles) colocando un llamado dentro del rol responsable de la tarea (Figura 18).

La mayor desventaja es que cada tarea colaborativa tendrá asociado un proceso, obligando a los modeladores a definir las propiedades y atributos que BPMN define.

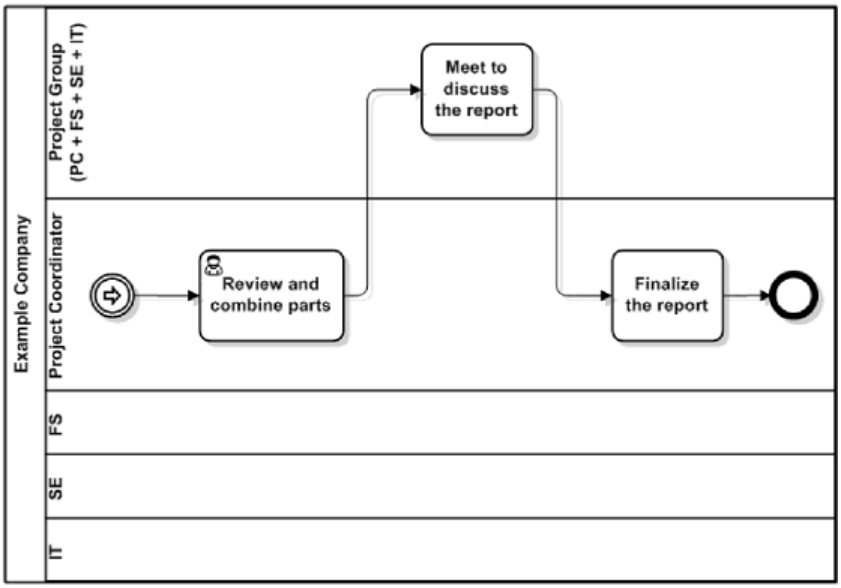


Figura 17: Ejemplo de carril separado [19]

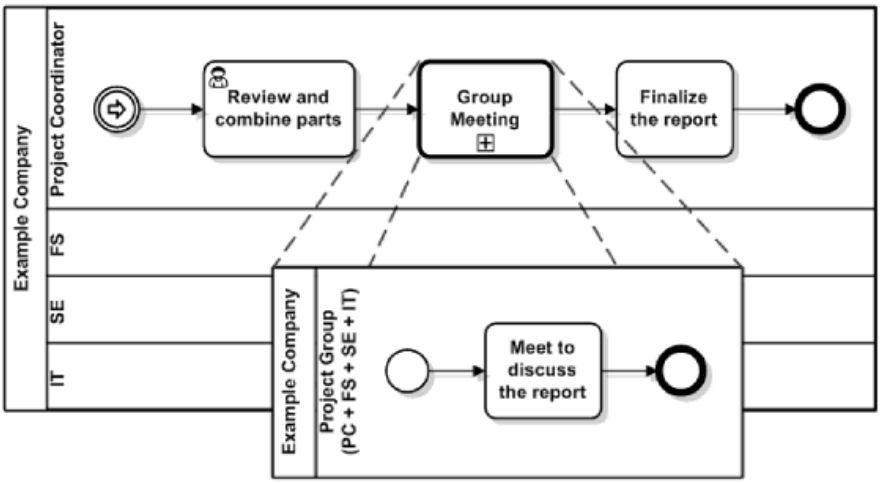


Figura 18: Ejemplo de nuevo proceso [19]

2.4 XSLT

Al igual que XML, XSLT es un lenguaje de programación. Forma parte de la trilogía transformadora de XML, compuesta por:

- CSS: (Cascading Style Sheets, hojas de estilo en cascada): Permite dar una apariencia en el navegador determinada a cada una de las etiquetas XML
- XSLT (XML Stylesheets Language for Transformation): Lenguaje de transformación basado en hojas de estilo)

- XSL:FO (Formatting Objects, objetos de formateo): o transformaciones para fotocomposición, o, en general, para cualquier cosa que no sea XML, como por ejemplo HTML o PDF

XSLT es pues, un lenguaje que se usa para convertir documentos XML en otros documentos XML. Puede convertir un documento XML a otro, o convertirlo a "formatos finales", tales como WML (usado en los móviles WAP) o XHTML (Figura 19).

Los programas XSLT están escritos en XML, y generalmente, se necesita un procesador de hojas de estilo, o Stylesheet Processor para procesarlas, aplicándolas a un archivo XML. El estilo de programación con las hojas XSLT es totalmente diferente a los otros lenguajes de programación declarativos, pareciéndose más a "lenguajes" tales como el AWK, o a otros lenguajes funcionales, tales como ML o Scheme. En la práctica, eso significa que una instrucción debe de hacer lo mismo cualquier que sea el camino de ejecución que llegue hasta ella. O sea, no va a haber variables, ni bucles en los que se incremente el valor de una variable, o tenga un test de fin de bucle, ni nada por el estilo. En realidad, esto no es tan grave, y se puede simular usando recursión.

Las hojas XSLT (en inglés se les llama Stylesheets o Logicsheets) son la única alternativa cuando uno quiere adaptar un contenido descrito con XML a diferentes clientes (por ejemplo, móviles de diferente tamaño, diferentes navegadores), y la mejor alternativa cuando uno quiere procesar documentos XML. Otra alternativa, sobre todo si se está trabajando ya con un documento XML en forma de DOM (Document Object Model) es trabajar directamente sobre él. En este caso, de todas formas, se pueden usar transformaciones XSL, sólo que se aplicarán en memoria, en vez de leerlas desde un archivo [24].

Lo que consiguen las hojas de estilo es separar la información (almacenada en un documento XML) de su presentación, usando en cada caso las transformaciones que sean necesarias para que el contenido aparezca de la forma más adecuada en el cliente. Es más, se pueden usar diferentes hojas de estilo, o incluso la misma, para presentar la información de diferentes maneras dependiendo de los deseos o de las condiciones del usuario.

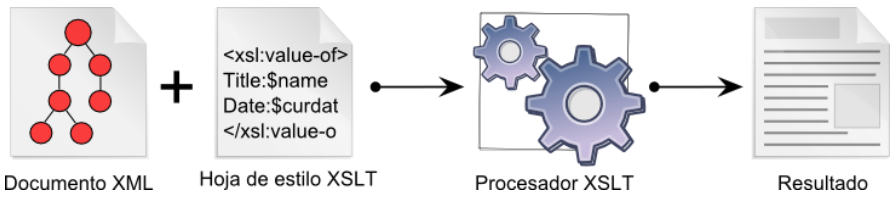


Figura 19: Esquema de Trabajo de XSLT

El lenguaje XSLT está normalizado por el W3C que ha publicado dos versiones de este lenguaje: XSLT 1.0 en noviembre de 1999 y XSLT 2.0 en enero de 2007 [25].

El uso de XSLT para la transformación de modelos es reconocido por diversos autores. En particular [26] indica que XSLT se usa como lenguaje para la transformación de

modelos en el espacio tecnológico[8] de XML (promovido por la W3C), mientras que otros lenguajes (como QVT) son apropiados para el espacio tecnológico MDA (basado en MOF y promovido por la OMG). Cabe destacar que tanto SPEM como BPMN están basados en MOF, por lo que son parte del espacio tecnológico MDA.

Por otro lado, [27] reporta el uso exitoso de XSLT para la transformación de modelos por su facilidad para la construcción de una prueba de concepto, pero que al evolucionar la herramienta a construir, debieron cambiar a un lenguaje más apropiado como QVT, debido a las limitaciones de XSLT.

Con respecto a las dificultades al usar XSLT para la transformación de modelos, [28] indica que XSLT tiene información de cálculo limitada en sus árboles, lo que implica que realizar varios recorridos en el árbol del modelo fuente (algo esencial para buscar la información necesaria para lograr la transformación de la entidad fuente) requiere la creación de árboles intermedios, lo que puede hacer que el proceso de transformación sea menos potente, difícil de leer y muy redundante. Por otro lado, [29] concluye que XSLT es adecuado para transformaciones sencillas pero presenta serias deficiencias para transformaciones más avanzadas, principalmente debido a que los documentos XML se representan como una estructura de árbol y los modelos son, en general, descriptibles naturalmente como gráficos, lo que puede conducir a una representación antinatural de muchos tipos de transformaciones de modelos.

2.5 Trabajo Relacionado

En la literatura disponible, se pueden encontrar algunas aproximaciones a la resolución de la problemática planteada, esto es la transformación de un proceso desde el metamodelo SPEM al metamodelo BPMN.

En 2009, [18] propone transformar la especificación en SPEM de la metodología de desarrollo SmallRUP a una especificación de procesos que pueda ser utilizada como entrada de un motor workflow estándar. Para ello, se utilizó una transformación definida en el lenguaje Relations que forma parte de Query/Views/Transformations (QVT). Dicha transformación convierte una especificación en SPEM en una especificación de procesos en BPMN. Dicha especificación, se transformará a su vez a una especificación BPEL. Sin embargo, los resultados sólo se limitaron a la transformación mediante QVT a BPMN, y actualmente no se encuentran disponibles.

En [20] 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 2012, [30] realiza un detallado análisis comparativo entre los metamodelos SPEM y BPMN. Se discuten criterios como su expresividad, reusabilidad, administración, evolución, comprensión e integración en la organización. Se incluye también una tabla

[8] Un espacio tecnológico es un contexto de trabajo con un conjunto de conceptos asociados, cuerpo de conocimientos, herramientas, habilidades requeridas y posibilidades. A menudo está asociada a una comunidad de usuarios con conocimientos compartidos, apoyo educativo, literatura común e incluso talleres y conferencias (*Ivan Kurtev, Jean Bézivin, Mehmet Aksit , Technological Spaces: an Initial Appraisal*)

en donde se presentan las correspondencias entre las entidades de los dos metamodelos.

También el año 2012 se presenta MOSKitt4ME [31] una herramienta cuyo desarrollo es liderado por la Universidad de Valencia en el marco del proyecto Modeling Software KIT (MOSKitt). 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 limitación es que no representa los roles responsables de las tareas en el BPMN obtenido.

En el año 2014, en [32] se realiza una transformación de SPEM a BPMN utilizando XSLT. Dicha transformación realiza una conversión parcial de elementos SPEM, dejando fuera algunos (Activity, Iteration, Phase, etc.). Dicha transformación tampoco se hace cargo de representar los roles en el modelo BPMN resultante, por lo que le es imposible representar tareas colaborativas.

3 Implementación

3.1 Reglas de Transformación de UMA/SPEM a BPMN

La realización de la transformación requiere definir un paralelismo entre los elementos de UMA/SPEM (utilizados en EPF Composer) y los elementos BPMN. Para ello se proponen las reglas contenidas en la Tabla 8. Es importante destacar que los elementos UMA/SPEM de la Tabla 8 corresponden a elementos *Activity* (diagramas de actividad) para la descripción de comportamiento de UML.

La aplicación de dichas reglas no considera ninguna modificación al flujo del proceso modelado en SPEM; en otras palabras, la idea es realizar una transformación de cada elemento SPEM a BPMN y mantener intacto el flujo. Por otro lado, se debe lidiar con la representación de roles en BPMN. Considerando que el objetivo de la presente investigación es facilitar la comunicación, se utilizarán los carriles para representar roles[9]. Sin embargo, esto supone un problema a la hora de representar tareas colaborativas. Según lo presentado en la sección 2.3.3, existen 4 alternativas para modelar este tipo de actividades:

1. Duplicar tareas en cada carril
2. Asignar diferentes tareas a los roles
3. Crear un carril para representar un grupo de roles
4. Crear un nuevo proceso para cada actividad colaborativa

Según [19] la solución 1 debe ser descartada (debido a los problemas que presenta, los que fueron descritos en la sección 2.3.3.1), siendo las mejores candidatas las soluciones 3 y 4. Para los propósitos de este trabajo, la solución 2 no es posible de automatizar debido a que la división de tareas debe ser realizada por el modelador, mientras que en [10], se recomienda utilizar la solución 3. Dado lo anterior, se ha decidido implementar las soluciones 3 y 4, lo que permitirá al modelador mayor flexibilidad a la hora de transformar su modelo SPEM a BPMN.

3.2 Transformación automática

La transformación automática tomará un modelo de proceso realizado en EPF Composer, el cual gracias a la división aportada por UMA a SPEM, guarda la información del Method Content en el archivo model.xmi y la información del flujo del proceso en el archivo diagram.xmi. Estos archivos serán recuperados y fusionados en el archivo root.xml, el cual contendrá toda la información necesaria para realizar la transformación. En la siguiente sección se presenta la definición del mapeo de nodos XML para los elementos y roles en SPEM a su respectivo elemento en BPMN.

[9] BPMN no especifica un significado para los carriles, dejándolo a criterio del modelador, quien puede utilizarlo para organizar y categorizar actividades dentro de un proceso. En la práctica, los modeladores entregan a los carriles un significado como por ejemplo: roles, sistemas, departamentos, etc. [19]

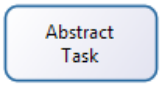
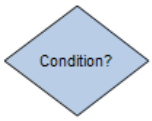
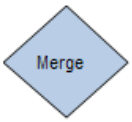
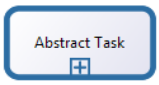
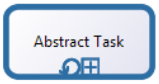
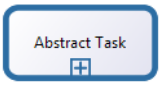
Descripción	Elemento SPEM/UMA	Elemento BPMN	Justificación
Start			Ambos cumplen la misma función, iniciar un proceso.
Task			Si bien es cierto que gran parte de las tareas SPEM/UMA son humanas (pudiendo considerarse una *Manual Task* en BPMN), se considera una aproximación más genérica, *Abstract Task.*
Decision			Un nodo de decisión SPEM/UMA toma un flujo de entrada y selecciona un flujo de salida. En BPMN un *Exclusive Gateway* tiene un comportamiento parecido (aunque puede recibir más de una entrada), por lo que se usa como su equivalente.
Fork, Join			En SPEM/UMA los elementos Fork y Join son usados para controlar flujos concurrentes. En BPMN se utiliza un *Parallel Gateway* tanto para dividir como para sincronizar flujos concurrentes, por lo que se usa como su equivalente.
Merge			Un nodo Merge (SPEM/UMA) recibe como entrada varios flujos alternativos (sin concurrencia) y permite una sola salida. En BPMN, un *Exclusive Gateway* tiene el mismo comportamiento, por lo que se usa como su equivalente.
Activity			Debido a que una actividad SPEM/UMA representa un conjunto de tareas, resulta adecuado llamar a un subproceso (*Call Activity* definido como *Abstract Task*) en BPMN.
Iteration			Una actividad iterativa SPEM/UMA representa un conjunto de tareas o actividades, por lo que resulta adecuado transformarla a una llamada a un subproceso (*Call Activity* definido como *Abstract Task*) en BPMN, marcada como *Loop.*
Phase			Dado que una fase SPEM/UMA representa un conjunto de tareas o actividades, resulta adecuado llamar a un subproceso (*Call Activity* definido como *Abstract Task*) en BPMN.
End			Ambos cumplen la misma función, finalizar un proceso.

Tabla 8: Tabla de Transformación de Elementos

3.2.1 Mapeo de Elementos SPEM a BPMN

El primer paso para la realización de la transformación es el mapeo de los elementos. La propuesta de correspondencia entre los elementos gráficos se aprecia en la Tabla 8.

Ahora corresponde automatizar dicha correspondencia. Para ello se procesa el archivo root.xml que describe el proceso definido en SPEM y se aplica una serie de transformaciones XPath para construir el modelo BPMN correspondiente.

La Tabla 9 muestra la transformación de nodos que automatiza la transformación. Es importante destacar que las actividades colaborativas se pueden transformar en dos tipos de nodos BPMN dependiendo del tratamiento que se desee entregarles (Carril Separado o Proceso Separado).

Descripción	SPEM (diagram//activity)	BPMN (definitions/process)
Start	node/@xmi:type="uml:InitialNode"	startEvent
Task, Tarea Colaborativa (usando Carriles Separados)	node/@xmi:type="uml:ActivityParameterNode"	Task
Task, Tarea Colaborativa (usando Procesos Separados)	node/@xmi:type="uml:ActivityParameterNode"	callActivity
Decision	node/@xmi:type="uml:DecisionNode"	exclusiveGateway
Fork, Join	node/@xmi:type="uml:ForkNode" node/@xmi:type="uml:JoinNode "	parallelGateway
Merge	node/@xmi:type="uml: MergeNode"	exclusiveGateway
Activity	node/@xmi:type="uml: StructuredActivityNode"	callActivity
Iteration	node/@xmi:type="uml: StructuredActivityNode"	callActivity (loop)
Phase	node/@xmi:type="uml: StructuredActivityNode"	callActivity
End	node/@xmi:type="uml: ActivityFinalNode"	endEvent
Sequence Flow	edge/@xmi:type="uml: ControlFlow"	sequenceFlow

Tabla 9: Tabla de Transformación de Nodos

3.2.2 Esquema de la Transformación

La transformación propuesta se realiza según lo indicado en la Figura 20. El flujo de la transformación es el siguiente:

1. Generar un modelo de procesos en EPF Composer.
2. Buscar los archivos del modelo generado (diagram.xmi y model.xmi) y unirlos en el archivo root.xml
3. Seleccionar un enfoque para el tratamiento de las tareas colaborativas: Carril Separado (Separated Lane) o Proceso Separado (Separated Process). Esta decisión implica la selección de un archivo XSLT que aplicará las reglas de la transformación.
4. Ejecutar la transformación. Esta transformación corresponde a la aplicación de la plantilla XSLT seleccionada en el paso anterior, al archivo root.xml. Esto puede realizarse con editores de texto (ej: Notepad++) y herramientas en línea (ej: http://www.utilities-online.info/xsltransformation/).
5. Guardar la transformación generada como un archivo con extensión bpmn (por ejemplo model.bpmn).
6. Abrir el archivo generado en un editor BPMN.

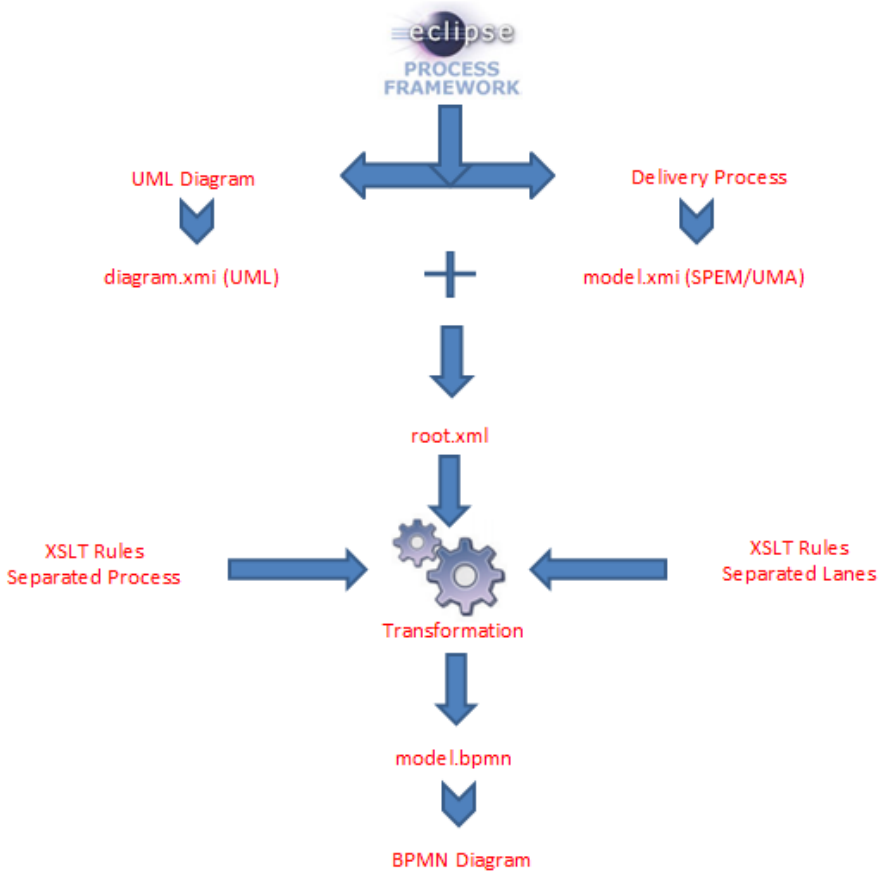


Figura 20: Estructura de la solución

A pesar de que la transformación puede realizarse de manera manual, para los propósitos de este trabajo se ha construido una herramienta de software, utilizando el Framework de programación .NET (Figura 21). Dicha herramienta permite buscar los modelos de software realizados en EPF Composer y transformarlos automáticamente a un archivo BPMN (automatizando los pasos 2, 3, 4 y 5). El modelo BPMN resultante puede ser cargado en algún editor de procesos (paso 6); en este caso se utiliza Bonita BPM[10].

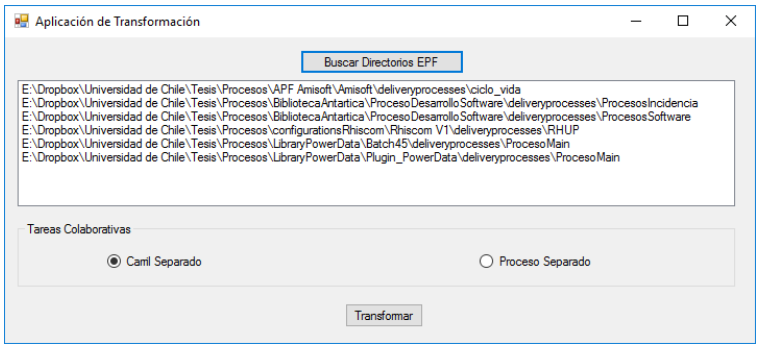


Figura 21: Herramienta de transformación

3.2.3 Estructura de los archivos de entrada y salida

Para ilustrar la estructura de los archivos de entrada involucrados en la transformación, se presenta el ejemplo de proceso generado en EPF Composer de la Figura 22.

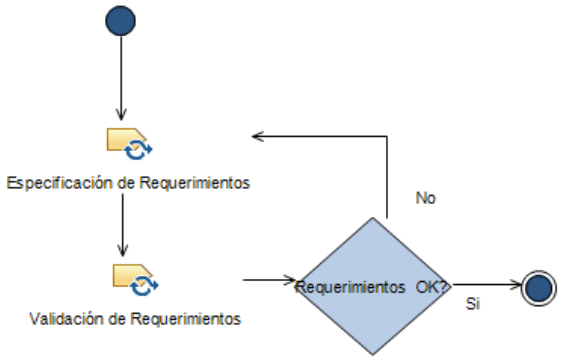


Figura 22: Ejemplo proceso generado en EPF Composer

Este proceso genera los dos archivos de entrada a la transformación:

[10] http://documentation.bonitasoft.com/

- model.xmi: Este archivo contiene la definición de la estructura en árbol de actividades, la secuencia entre ellas y el uso de roles y artefactos. En otras palabras, guarda la definición de un DeliveryProcess en UMA (Figura 23). De particular importancia para la transformación son los nodos processElements que contiene nombre de presentación (atributo presentationName) de los elementos del modelo y los roles responsables de cada tarea (atributo performedPrimarilyBy).

Figura 23: Ejemplo formato de archivo model.xmi

- diagram.xmi: Contiene la definición de los diagramas de actividad de UML que se asocian potencialmente a cada una de las actividades de las estructura del proceso. Destaca en este archivo el uso de elementos pertenecientes al Activity de UML (Figura 24). De particular importancia para la transformación son los nodos Node, que representan un elemento, según lo indicado en el parámetro xmi:type. Este parámetro puede tener diversos valores (uml:ActivityParameterNode para tareas, uml:InitialNode para nodos de inicio, uml:DecisionNode para decisiones, uml:ControlFlow para flechas de flujo, etc.)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.0" 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">
  <uml:Activity xmi:id="_Wq-_YB7EEee1y5RqytEGTA" name="Requirement_Capture">
    <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Wq-_YR7EEee1y5RqytEGTA" source="uma_element">
      <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Wq-_Yh7EEee1y5RqytEGTA" key="uri"
        value="uma://_IMUWsGsoEeaE8Y4Q6oK1AQ#_IMUWsWsoEeaE8Y4Q6oK1AQ"/>
      <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Wq-_Yx7EEee1y5RqytEGTA" key="type" value="Activity"/>
    </eAnnotations>
    <node xmi:type="uml:ActivityParameterNode" xmi:id="_Wq-_aB7EEee1y5RqytEGTA" name="Especificación de Requerimientos"
      outgoing="_Wq-_dh7EEee1y5RqytEGTA" incoming="_Wq-_ex7EEee1y5RqytEGTA _E2f_MB7GEee9W69ZuwbEDg">
      <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Wq-_aR7EEee1y5RqytEGTA" source="uma_element">
        <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Wq-_ah7EEee1y5RqytEGTA" key="uri"
          value="uma://_IMUWsGsoEeaE8Y4Q6oK1AQ#_LuYvQGsoEeaE8Y4Q6oK1AQ"/>
        <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Wq-_ax7EEee1y5RqytEGTA" key="type" value="Task"/>
      </eAnnotations>
    </node>
    <node xmi:type="uml:ActivityParameterNode" xmi:id="_Wq-_bB7EEee1y5RqytEGTA" name="Validación de Requerimientos"
      outgoing="_Wq-_dx7EEee1y5RqytEGTA" incoming="_Wq-_dh7EEee1y5RqytEGTA">
      <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Wq-_bR7EEee1y5RqytEGTA" source="uma_element">
        <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Wq-_bh7EEee1y5RqytEGTA" key="uri"
          value="uma://_IMUWsGsoEeaE8Y4Q6oK1AQ#_MJ18MGsoEeaE8Y4Q6oK1AQ"/>
        <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Wq-_bx7EEee1y5RqytEGTA" key="type" value="Task"/>
      </eAnnotations>
    </node>
    <node xmi:type="uml:InitialNode" xmi:id="_Wq-_cB7EEee1y5RqytEGTA" outgoing="_E2f_MB7GEee9W69ZuwbEDg"/>
    <node xmi:type="uml:ActivityFinalNode" xmi:id="_Wq-_cR7EEee1y5RqytEGTA" incoming="_Wq-_eR7EEee1y5RqytEGTA"/>
    <node xmi:type="uml:DecisionNode" xmi:id="_Wq-_ch7EEee1y5RqytEGTA" name="Requerimientos  OK?"
      outgoing="_Wq-_eR7EEee1y5RqytEGTA _Wq-_ex7EEee1y5RqytEGTA" incoming="_Wq-_dx7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_cx7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_dB7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_dR7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_dh7EEee1y5RqytEGTA" source="_Wq-_aB7EEee1y5RqytEGTA"
      target="_Wq-_bB7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_dx7EEee1y5RqytEGTA" source="_Wq-_bB7EEee1y5RqytEGTA"
      target="_Wq-_ch7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_eB7EEee1y5RqytEGTA" name="No"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_eR7EEee1y5RqytEGTA" name="Si" source="_Wq-_ch7EEee1y5RqytEGTA"
      target="_Wq-_cR7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_eh7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_Wq-_ex7EEee1y5RqytEGTA" name="No" source="_Wq-_ch7EEee1y5RqytEGTA"
      target="_Wq-_aB7EEee1y5RqytEGTA"/>
    <edge xmi:type="uml:ControlFlow" xmi:id="_E2f_MB7GEee9W69ZuwbEDg" source="_Wq-_cB7EEee1y5RqytEGTA"
      target="_Wq-_aB7EEee1y5RqytEGTA"/>
  </uml:Activity>
</xmi:XMI>
```

Figura 24: Ejemplo formato archivo diagram.xmi

Los dos archivos anteriormente mencionados deben ser fusionados en un solo archivo (root.xml) que es el que alimentará la transformación automática. Este archivo debe tener la estructura mostrada en la Figura 25.

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <root>
        <diagram>
            ... Contenido del archivo diagram.xmi ...
        </diagram>
        <model>
            ... Contenido del archivo model.xmi ...
        </model>
    </root>
```

Figura 25: Ejemplo formato archivo root.xml

La trasformación automática deberá generar un archivo con extensión bpmn que contendrá el proceso definido en BPMN. Destacan en este archivo el nodo process (que permite definir un proceso), el nodo laneSet (donde se definen un nodo lane por cada carril del proceso), el nodo lane (donde se coloca un nodo flowNodeRef con el respectivo identificador para cada elemento contenido en el carril), el nodo task (que identifica una tarea), y los nodos sequenceFlow (que definen las flechas de flujo entre los elementos).

```
<definitions 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"
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI">
    <process name="Requirement_Capture" id="_Wq-_YB7EEee1y5RqytEGTA">
        <laneSet>
            <lane name="Desarrollador">
                <flowNodeRef>_Wq-_aB7EEee1y5RqytEGTA</flowNodeRef>
                <flowNodeRef>_Wq-_cB7EEee1y5RqytEGTA</flowNodeRef>
                <flowNodeRef>_Wq-_cR7EEee1y5RqytEGTA</flowNodeRef>
                <flowNodeRef>_Wq-_ch7EEee1y5RqytEGTA</flowNodeRef>
            </lane>
            <lane name="Analista de Negocios + Jefe de Proyecto">
                <flowNodeRef>_Wq-_bB7EEee1y5RqytEGTA</flowNodeRef>
            </lane>
        </laneSet>
        <task completionQuantity="1" isForCompensation="false" startQuantity="1" id="_Wq-_aB7EEee1y5RqytEGTA"
            name="Especificación de Requerimientos">
            <incoming>_Wq-_ex7EEee1y5RqytEGTA</incoming>
            <incoming>_E2f_MB7GEee9W69ZuwbEDg</incoming>
            <outgoing>_Wq-_dh7EEee1y5RqytEGTA</outgoing>
        </task>
        <task completionQuantity="1" isForCompensation="false" startQuantity="1" id="_Wq-_bB7EEee1y5RqytEGTA"
            name="Validación de Requerimientos">
            <incoming>_Wq-_dh7EEee1y5RqytEGTA</incoming>
            <outgoing>_Wq-_dx7EEee1y5RqytEGTA</outgoing>
        </task>
        <startEvent id="_Wq-_cB7EEee1y5RqytEGTA" name="Inicio">
            <outgoing>_E2f_MB7GEee9W69ZuwbEDg</outgoing>
        </startEvent>
        <endEvent id="_Wq-_cR7EEee1y5RqytEGTA" name="Fin">
            <incoming>_Wq-_eR7EEee1y5RqytEGTA</incoming>
        </endEvent>
        <exclusiveGateway gatewayDirection="Diverging" id="_Wq-_ch7EEee1y5RqytEGTA" name="Requerimientos  OK?">
            <incoming>_Wq-_dx7EEee1y5RqytEGTA</incoming>
            <outgoing>_Wq-_eR7EEee1y5RqytEGTA</outgoing>
            <outgoing>_Wq-_ex7EEee1y5RqytEGTA</outgoing>
        </exclusiveGateway>
        <sequenceFlow id="_Wq-_dh7EEee1y5RqytEGTA" name="" sourceRef="_Wq-_aB7EEee1y5RqytEGTA"
            targetRef="_Wq-_bB7EEee1y5RqytEGTA" />
        <sequenceFlow id="_Wq-_dx7EEee1y5RqytEGTA" name="" sourceRef="_Wq-_bB7EEee1y5RqytEGTA"
            targetRef="_Wq-_ch7EEee1y5RqytEGTA" />
        <sequenceFlow id="_Wq-_eR7EEee1y5RqytEGTA" name="Si" sourceRef="_Wq-_ch7EEee1y5RqytEGTA"
            targetRef="_Wq-_cR7EEee1y5RqytEGTA" />
        <sequenceFlow id="_Wq-_ex7EEee1y5RqytEGTA" name="No" sourceRef="_Wq-_ch7EEee1y5RqytEGTA"
            targetRef="_Wq-_aB7EEee1y5RqytEGTA" />
        <sequenceFlow id="_E2f_MB7GEee9W69ZuwbEDg" name="" sourceRef="_Wq-_cB7EEee1y5RqytEGTA"
            targetRef="_Wq-_aB7EEee1y5RqytEGTA" />
    </process>
</definitions>
```

Figura 26: Ejemplo formato archivo bpmn

3.2.4 Mapeo de Roles con Carril Separado

En este caso el mapeo se realiza recorriendo las tareas (nodos Activity) definidos en el proceso (nodo Diagram). Una vez que se identifica una tarea, se procede a buscar en su definición (nodo Model) los roles marcados como responsables primarios, para crear un carril (nodo LaneSet) para BPMN. Luego se procede a asignar las tareas al carril correspondiente. La mayor dificultad de este procedimiento es realizar la agrupación de

las líneas de secuencia en el carril correspondiente. En la Figura 27 se puede apreciar el algoritmo de la solución, el cual será implementado en una plantilla XSLT (XSLT Rules – Separated Lane)

1: Crear un nodo ⟨*definitions*⟩
2: **for all** *spemActivity* ← ⟨*uml* : *Activity*⟩ en ⟨*diagram*⟩ **do**
3: Crear un nodo ⟨*process*⟩ en ⟨*definitions*⟩
4: Asignar los atributos de ⟨*process*⟩ (id, name) con los valores de *spemActivity* en ⟨*model*⟩
5: Crear un nodo ⟨*laneSet*⟩ en ⟨*process*⟩
6: **for all** *spemTask* ← ⟨*node*[@*xmi* : *type* =′ *uml* : *ActivityParameterNode*′]⟩ en *spemActivity* **do**
7: *spemRole* ← **rol(es)** de *spemTask* definidos en ⟨*model*⟩
8: **if** *spemRole* está vacío **then**
9: Crear un nodo ⟨*lane*⟩ en ⟨*laneSet*⟩, asignando su atributo name como *Undefined*
10: **else**
11: Crear un nodo ⟨*lane*⟩ en ⟨*laneSet*⟩, asignando su atributo name igual al atributo name de *spemRole*
12: **end if**
13: Crear un nodo ⟨*flowNodeRef*⟩ en ⟨*lane*⟩, asignando su valor igual al atributo id de *spemTask*
14: **end for**
15: Agrupar los nodos ⟨*flowNodeRef*⟩ que tengan el mismo nodo ⟨*lane*⟩
16: Crear los elementos (según lo indicado en la Tabla de Transformación de Nodos) agregándolos al nodo ⟨*process*⟩
17: **end for**
18: **return** Nodo ⟨*definitions*⟩

Figura 27: Algoritmo Carril Separado

3.2.5 Mapeo de Roles con Proceso Separado

La primera parte del mapeo es idéntico al anterior. La diferencia consiste en que las tareas que tiene más de un responsable son mapeadas como un callActivity en BPMN. Finalmente, la transformación crea los procesos adicionales (con el prefijo Meeting), creando un nuevo proceso para las tareas que tienen más de un responsable.

En la Figura 28 se puede apreciar el algoritmo de la solución, el cual será implementado en una plantilla XSLT (XSLT Rules – Separated Process)

1: Crear un nodo ⟨*definitions*⟩
2: **for all** *spemActivity* ← ⟨*uml* : *Activity*⟩ en ⟨*diagram*⟩ **do**
3: Crear un nodo ⟨*process*⟩ en ⟨*definitions*⟩
4: Asignar los atributos de ⟨*process*⟩ (id, name) con los valores de *spemActivity* en ⟨*model*⟩
5: Crear un nodo ⟨*laneSet*⟩ en ⟨*process*⟩
6: **for all** *spemTask* ← ⟨*node*[@*xmi* : *type* =′ *uml* : *ActivityParameterNode*′]⟩ en *spemActivity* **do**
7: *spemRole* ← **primer rol** de *spemTask* definido en ⟨*model*⟩
8: **if** *spemRole* está vacío **then**
9: Crear un nodo ⟨*lane*⟩ en ⟨*laneSet*⟩, asignando su atributo name como *Undefined*
10: **else**
11: Crear un nodo ⟨*lane*⟩ en ⟨*laneSet*⟩, asignando su atributo name igual al atributo name de *spemRole*
12: **end if**
13: Crear un nodo ⟨*flowNodeRef*⟩ en ⟨*lane*⟩, asignando su valor igual al atributo id de *spemTask*
14: **end for**
15: Agrupar los nodos ⟨*flowNodeRef*⟩ que tengan el mismo nodo ⟨*lane*⟩
16: Crear los elementos (según lo indicado en la Tabla de Transformación de Nodos) agregándolos al nodo ⟨*process*⟩
17: **end for**
18: **for all** *spemTask* ← ⟨*node*[@*xmi* : *type* =′ *uml* : *ActivityParameterNode*′]⟩ en ⟨*diagram*⟩ **do**
19: *spemRole* ← **rol(es)** de *spemTask* definidos en ⟨*model*⟩
20: **if** *spemRole* tiene más de un elemento **then**
21: Crear un nodo ⟨*process*⟩ en ⟨*definitions*⟩
22: Asignar los atributos de ⟨*process*⟩ (nuevo id; name como la concatenación de 'Meeting' y los atributos name de *spemRole*)
23: Crear un nodo ⟨*laneSet*⟩ en ⟨*process*⟩
24: Crear un nodo ⟨*lane*⟩ en ⟨*laneSet*⟩, asignando su atributo name igual a los atributos name de *spemRole*
25: Crear una tarea, un nodo de inicio y un nodo de fin
26: Crear tres nodos ⟨*flowNodeRef*⟩ en ⟨*lane*⟩, asignando los valores de los ID de la nueva tarea y nodos de inicio y fin
27: **end if**
28: **end for**
29: **return** Nodo ⟨*definitions*⟩

Figura 28: Algoritmo Proceso Separado

3.2.6 Limitaciones

- Debido a que no poseen un rol definido, los nodos de inicio, fin, decisión, Fork, Join, Merge, Activity e Iteration de un proceso son asignados al primer rol que aparezca en el proceso.
- Las tareas que no tengan un responsable principal definido en SPEM, aparecen en BPMN asignadas al carril "Undefined".

- SPEM permite definir un elemento de tipo “Milestone” que no ha sido transformado a BPMN debido a que no se ha encontrado un equivalente desde la perspectiva de la definición de procesos.
- La transformación no considera la inclusión de los elementos adicionales a las tareas del proceso (Guidances y WorkProducts), principalmente porque esto agrega demasiada complejidad a las reglas de transformación, escapando al alcance propuesto para el presente trabajo.
- El archivo resultante no incluye información respecto a la disposición gráfica de los elementos BPMN, por lo que el diagrama resultante debe ser ordenado para su correcta visualización.

4 Validación

A estas alturas, se tiene un modelo de proceso definido en BPMN obtenido mediante la transformación de un modelo de proceso de software definido en SPEM. Sin embargo, dicha transformación debe ser validada en términos del formato BPMN obtenido (validación sintáctica) y en términos de la equivalencia del modelo obtenido con el original (validación semántica). Ambas validaciones se realizarán utilizando un caso de estudio de la vida real.

Con respecto a la validación sintáctica, cabe destacar que durante esta investigación se ha utilizado Bonita BPM, herramienta que permite la importación de modelos en formato BPMN 2.0 [23]. Si se considera que dicha herramienta puede abrir el modelo BPMN generado por la transformación automática sin problemas, la validación del formato generado resulta exitosa. Sin embargo, también se debe validar que la transformación puede manejar correctamente todos los elementos SPEM incluidos en la tabla de transformaciones (Tabla 8). El caso de estudio planteado cubre 8 de las 9 posibles transformaciones en distintas combinaciones, dejando fuera solamente el caso del elemento Iteration. El elemento Iteration fue testeado exitosamente con otros modelos de procesos de software obtenidos en el laboratorio, por lo que es posible indicar que su comportamiento es el esperado. Esto, sumado a los resultados del caso de estudio, indica que la transformación resulta ser sintácticamente correcta.

En el caso de la validación semántica, el caso de estudio debe responder la siguiente pregunta: ¿es el proceso BPMN resultante equivalente al proceso original definido en SPEM? Para responder dicha pregunta en las siguientes secciones se aplica la transformación a procesos de software reales (secciones 4.1 y 4.2) y se discuten los resultados (sección 4.3).

4.1 Caso de Estudio

Para el presente trabajo se utilizan como caso de estudio los procesos de software utilizados en Librería Antártica[11] para sus sistemas de ventas, contabilidad y abastecimientos.

Librería Antártica posee 19 tiendas en el país distribuidas en las ciudades de Antofagasta, Concepción, Viña del Mar, Temuco y Santiago, además de una tienda virtual en Internet.

En el año 2014, el Departamento de Sistemas de Librería Antártica formaliza, utilizando EPF Composer, los procesos de Incidencias y Nuevos Desarrollos. Esta formalización resulta en el plug-in “ProcesoDesarrolloSoftware” (Figura 29).

Desde la perspectiva de los procesos, el plug-in contiene un Capability Pattern (cp_PuestaEnMarcha) y dos Delivery Process (ProcesosIncidencia y ProcesosSoftware).

[11] http://www.antartica.cl/antartica/index.jsp

Cabe mencionar que durante la construcción de la transformación, se utilizó el proceso de Incidencias (Delivery Process ProcesosIncidencia) para la experimentación de las alternativas de representación de las tareas colaborativas.

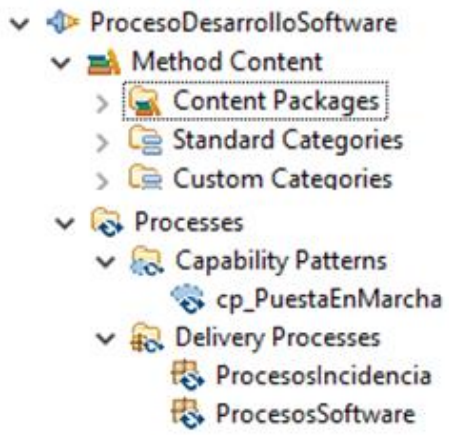


Figura 29: Vista General Plug-in Antartica

Dentro del Method Content del plug-in, se puede encontrar la definición de los roles, guías, tareas y productos de trabajo (Figura 30 y Figura 31)

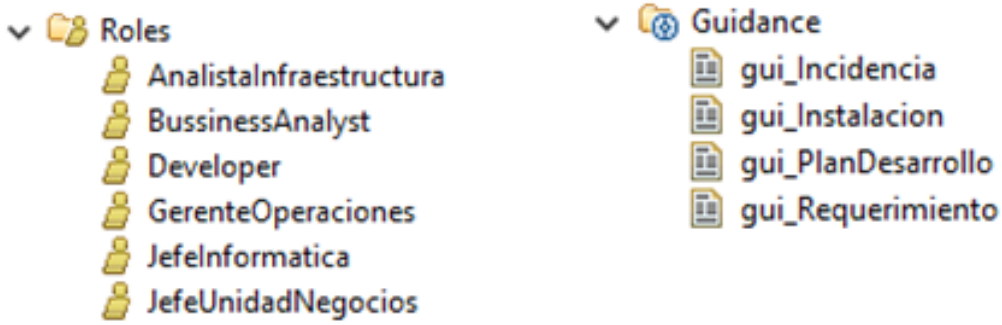


Figura 30: Roles y Guías

El Capability Pattern “cp_PuestaEnMarcha” (Figura 32) describe el proceso de instalación de una pieza de software, ya sea nueva o de resolución de incidencia. Se define como un patrón para ser reutilizado en diferentes procesos. En particular se usa tanto en el proceso de nuevos desarrollos como en el de incidencias.

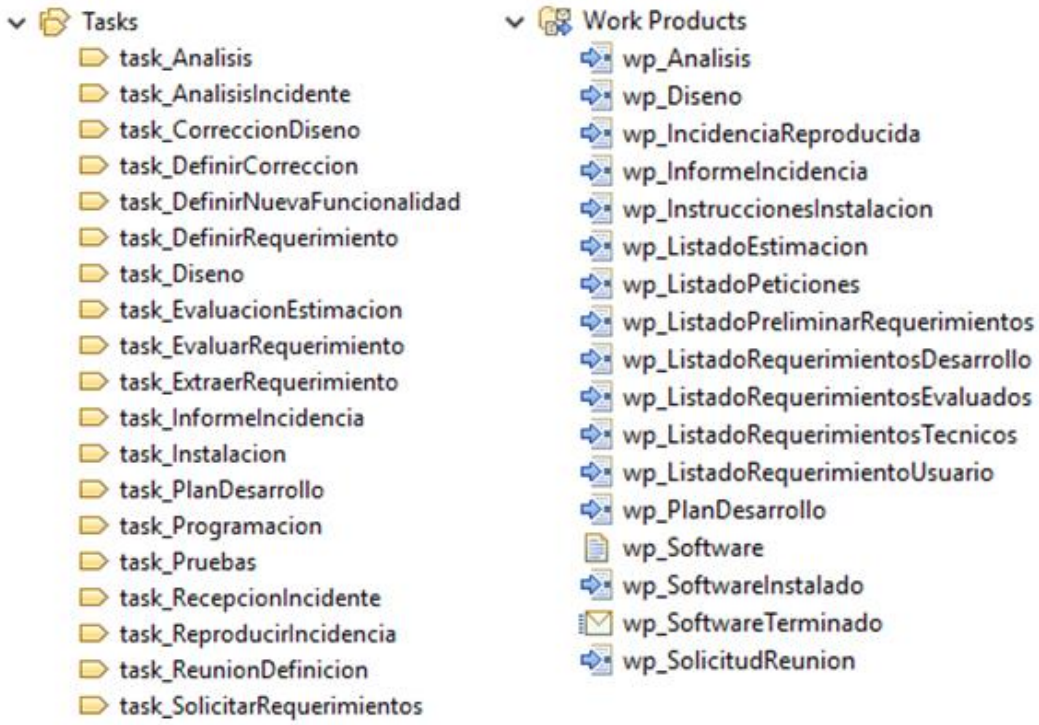


Figura 31: Tareas y Productos de Trabajo

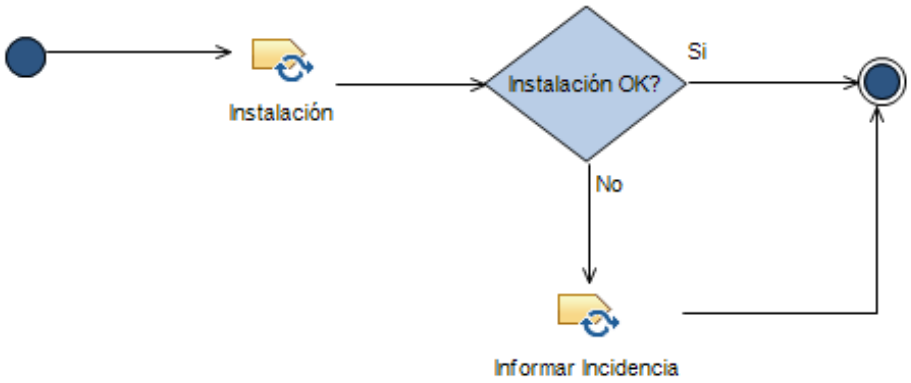


Figura 32: Capability Pattern "cp_PuestaEnMarcha"

En el Delivery Process "ProcesosIncidencia" (Figura 33) se describe el proceso de resolución de incidencias, el cual está compuesto de tres fases: Detección (Figura 34), Construir Solución (Figura 35) y Puesta en Marcha (mediante el Capability Pattern "PuestaEnMarcha").

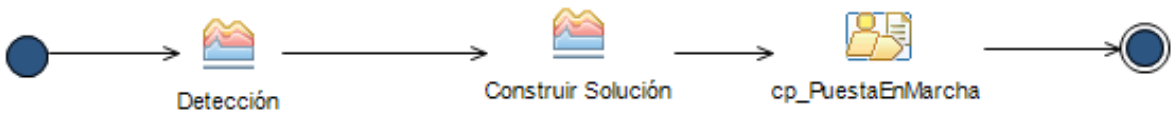


Figura 33: Vista General ProcesosIncidencia

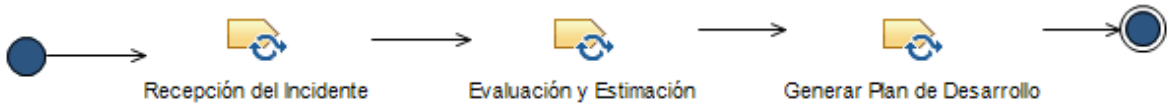


Figura 34: ProcesosIncidencia – Detección

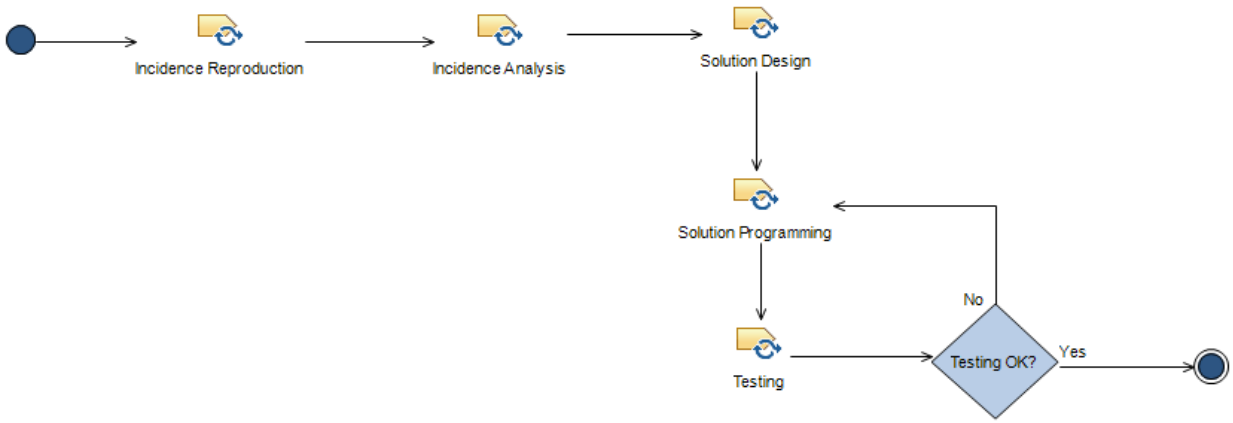


Figura 35: ProcesosIncidencia – Construir Solución

En el Delivery Process “ProcesosSoftware” (Figura 36) se describe el proceso de creación de nuevas piezas de software o modificaciones a las ya existentes. Dicho proceso está compuesto de tres fases: Definición del Proyecto (Figura 37), Desarrollo (Figura 38) y Puesta en Marcha (mediante el Capability Pattern “PuestaEnMarcha”).

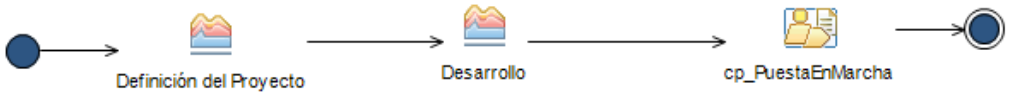


Figura 36: Vista General ProcesosSoftware

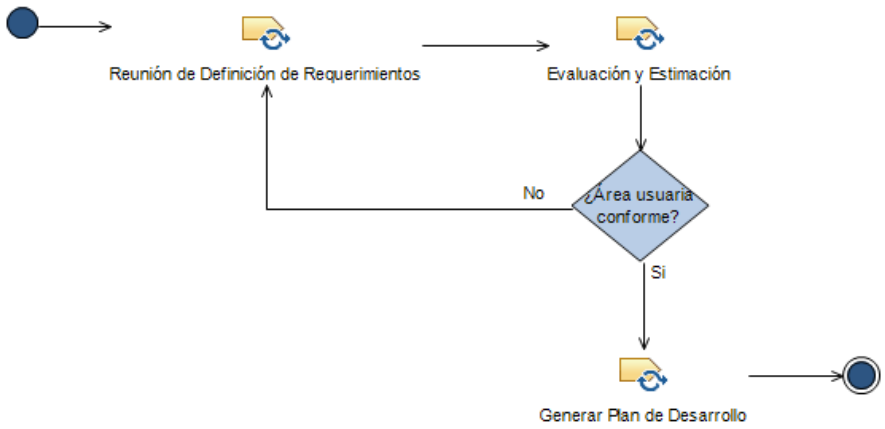


Figura 37: ProcesosSoftware – Definición del Proyecto

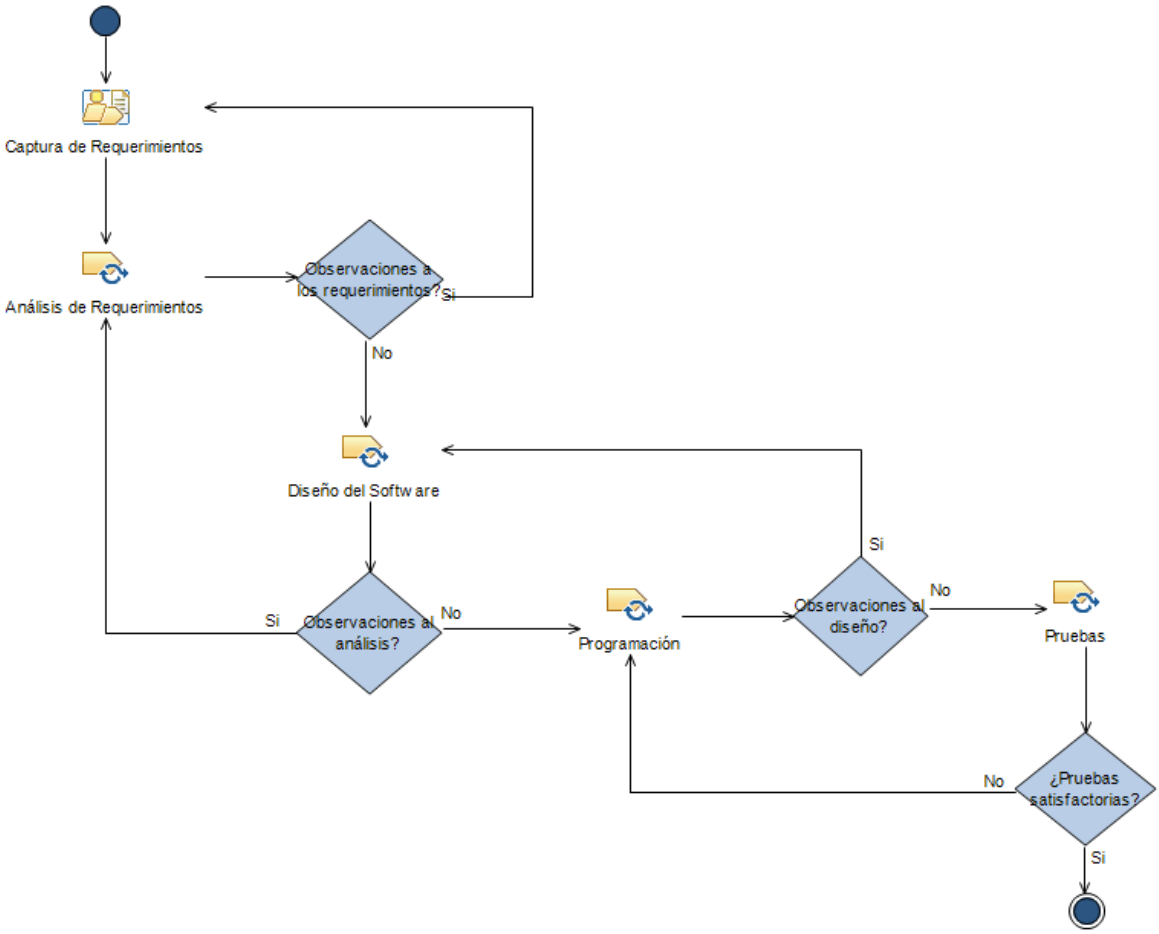


Figura 38: ProcesosSoftware – Desarrollo

Destaca en la fase de Desarrollo la aplicación de un Activity (Captura de Requerimientos), el cual tiene como objetivo agrupar las tareas de la captura de requerimientos (Figura 39). Dentro de él se puede apreciar el uso de un Fork y un Join,

los cuales permiten sincronizar las actividades contenidas entre ellas. Este Activity permitirá validar el uso de este tipo de compuertas.

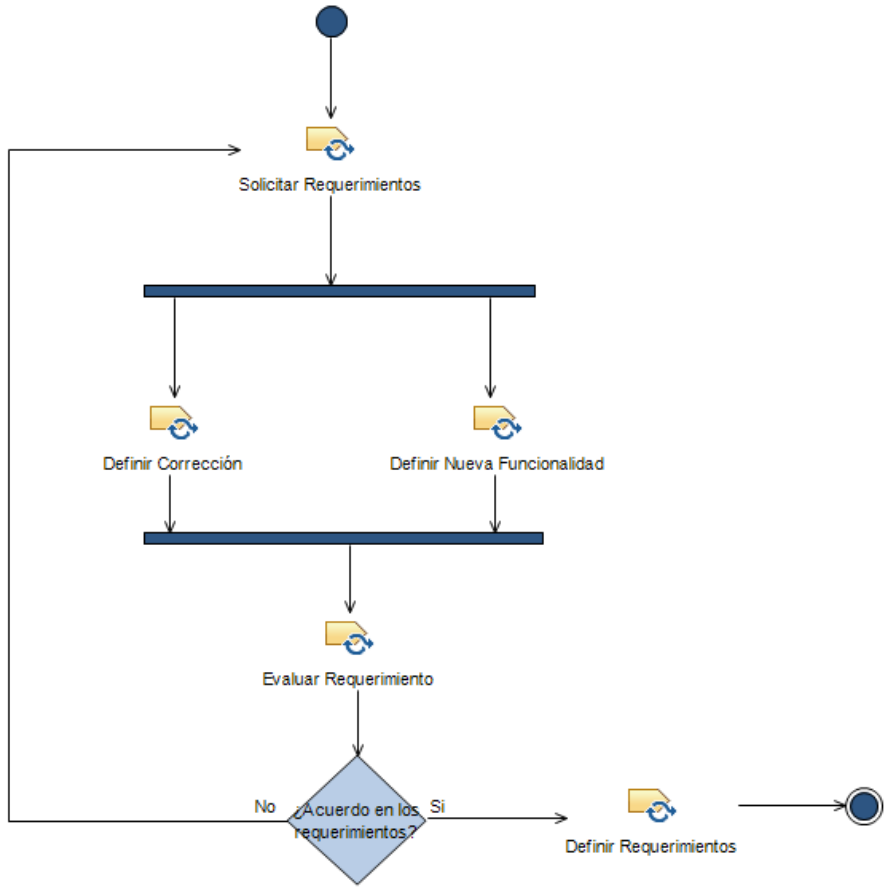


Figura 39: ProcesosSoftware - Desarrollo - Captura de Requerimientos

Para los objetivos del presente trabajo, se deben identificar las tareas colaborativas. Cabe recordar que en SPEM, dicha definición se realiza en el Method Content, al definir la tarea, al momento de asignar sus roles (Figura 40).

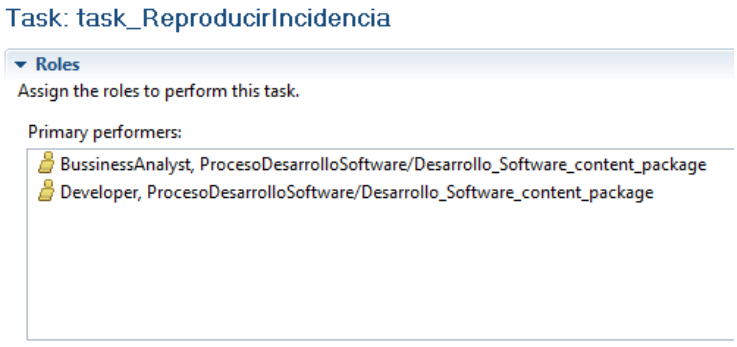


Figura 40: Asignación de roles en SPEM

Dado lo anterior, siete de las tareas definidas en los procesos de Antártica corresponden a tareas colaborativas. Los responsables de cada tarea pueden verse en la Tabla 10 donde las tareas colaborativas están resaltadas en rojo.

Tarea	Analista Infraestr.	Analista Negocios	Desarrolla-dor	Gerente Operaciones	Jefe Informática	Jefe Unidad Negocios
Análisis			X			
Análisis Incidente			X			
Corrección Diseño			X			
Definir Corrección			X			
Definir Nueva Funcionalidad			X			
Definir Requerimiento			X			
Diseño			X			
Evaluación y Estimación				X		
Evaluar Requerimiento		X			X	
Extraer Requerimiento		X			X	
Informar Incidencia			X			
Instalación	X					
Plan Desarrollo					X	
Programación			X			
Pruebas		X	X			
Recepción Incidente			X		X	
Reproducir Incidencia		X	X			
Reunión Definición				X		X
Solicitar Requerimientos		X			X	

Tabla 10: Responsables de cada tarea

4.2 Transformación del Caso de Estudio

Se aplica la transformación a los dos procesos del caso de estudio. Las actividades colaborativas serán tratadas de forma independiente con las dos opciones de tratamiento: Carril Separado y Proceso Separado. El archivo BPMN resultante en cada caso es abierto utilizando Bonita BPM.

4.2.1 Carril Separado

Al aplicar la transformación al Proceso de Incidencias, se puede apreciar la misma distribución que en SPEM en el diagrama resultante. Al corresponder todas a fases o patrones de proceso y no haber ninguna tarea, no aparecen carriles asociados a roles. Destaca la transformación de fases y patrones de proceso en llamados a sub-procesos (Figura 41).

Adicionalmente, este diagrama es idéntico al generado por la transformación automática en el caso del proceso separado (Figura 50), ya que no cuenta con roles que hagan la diferencia en el tratamiento de carriles o actividades.

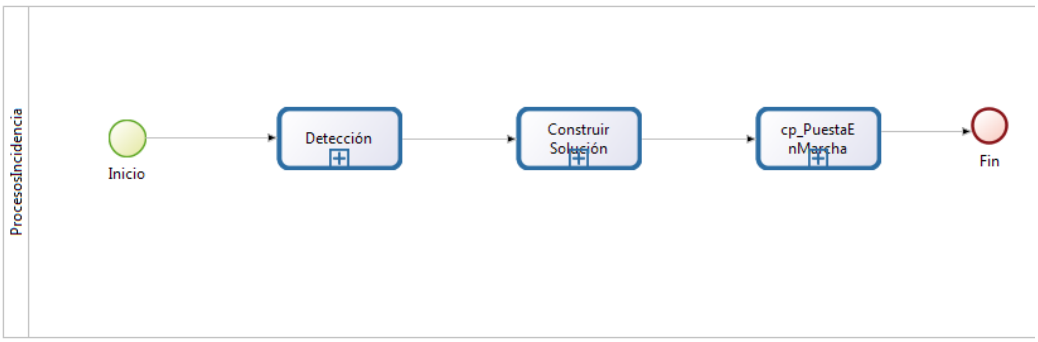


Figura 41: Diagrama General ProcesosIncidencia – BPMN Carril Separado

La transformación automática modela el subproceso “Detección” utilizando tres carriles, ya que además de las tareas realizadas exclusivamente por el Jefe de Informática y el Gerente de Operaciones, aparece un carril asociado la reunión del Desarrollador con el Jefe de Informática.

Cabe destacar que los nodos de inicio y fin aparecen en el carril compuesto (Figura 42), aunque esto es sólo una decisión arbitraria.

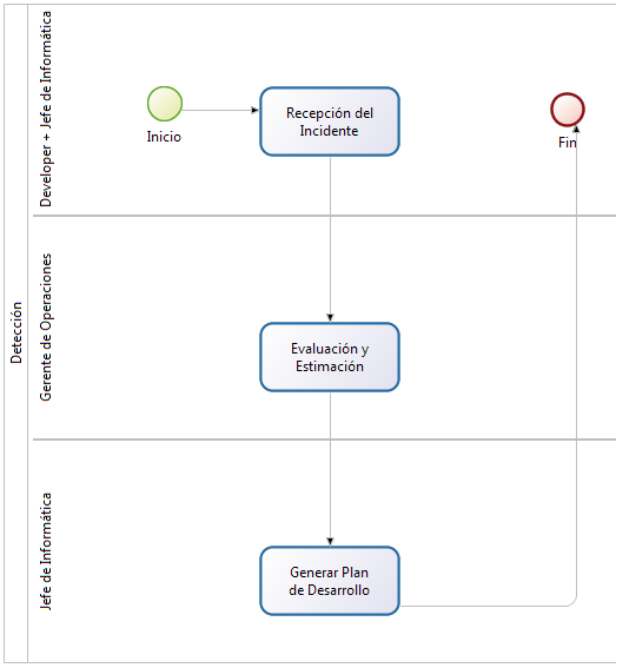


Figura 42: Diagrama Detección - ProcesosIncidencia – BPMN Carril Separado

La transformación modela el subproceso de “Construir Solución” utilizando dos carriles, uno para las tareas exclusivas del desarrollador, y otro para las tareas que debe realizar en conjunto con el Analista de Negocios. En este diagrama se puede apreciar que la compuerta de decisión aparece asignada al carril asociado al rol compuesto (Figura 43).

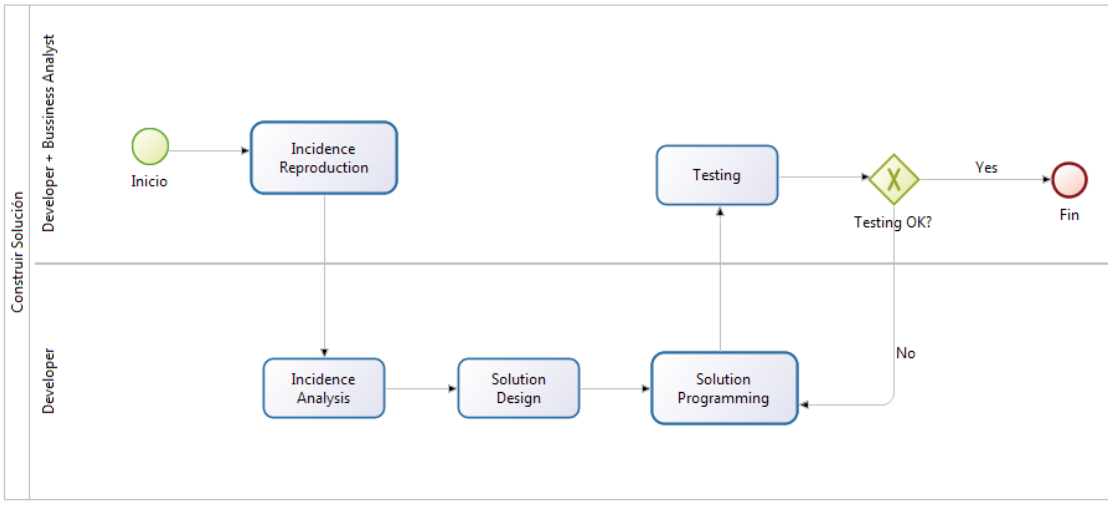


Figura 43: Diagrama Construir Solución - ProcesosIncidencia – BPMN Carril Separado

El subproceso de “cp_PuestaEnMarcha” corresponde al modelado del patrón de proceso del mismo nombre en SPEM. En este modelado no existen tareas

colaborativas, por lo que aparecen sólo dos carriles, uno por cada rol involucrado (Figura 44).

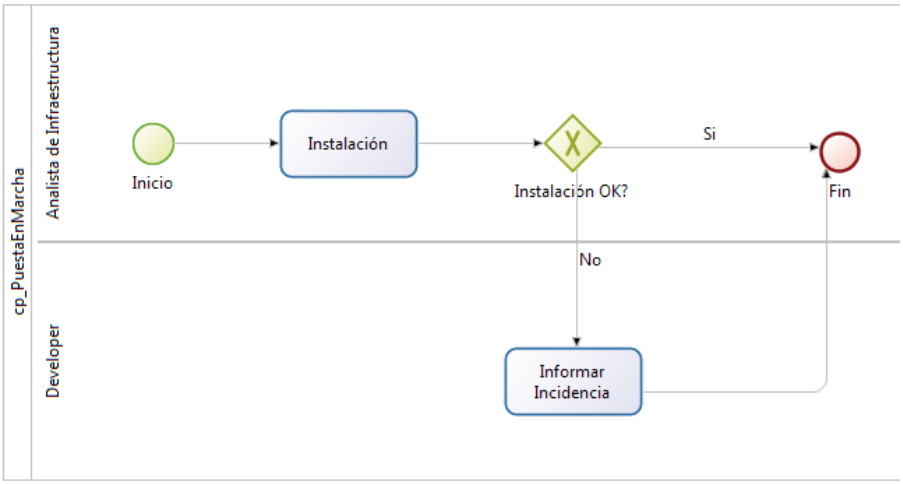


Figura 44: Diagrama cp_PuestaEnMarcha - ProcesosIncidencia – BPMN Carril Separado

Por otra parte, al aplicar la transformación al Proceso de Desarrollo de Software, en su diagrama general BPMN (Figura 45) no se observan mayores diferencias respecto al diagrama SPEM.

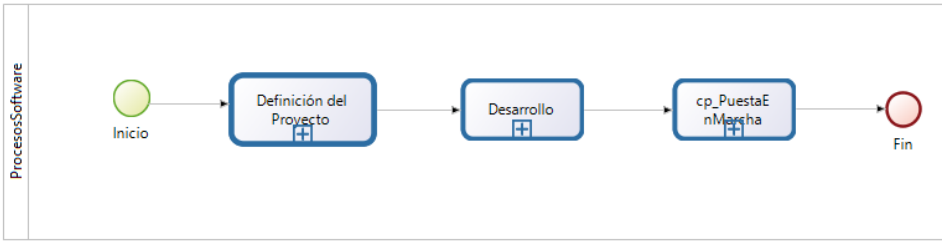


Figura 45: Diagrama General ProcesosSoftware – BPMN Carril Separado

En el subproceso “Definición del Proyecto” se observan tres carriles, los dos últimos corresponden a roles definidos en SPEM, mientras que el primero ha sido generado por la transformación para reflejar el trabajo conjunto de dos roles al realizar la tarea “Reunión de Definición de Requerimientos” (Figura 46).

También se puede observar que la transformación del patrón de proceso “cp_PuestaEnMarcha” es idéntico al caso anterior, manteniendo la coherencia (Figura 47).

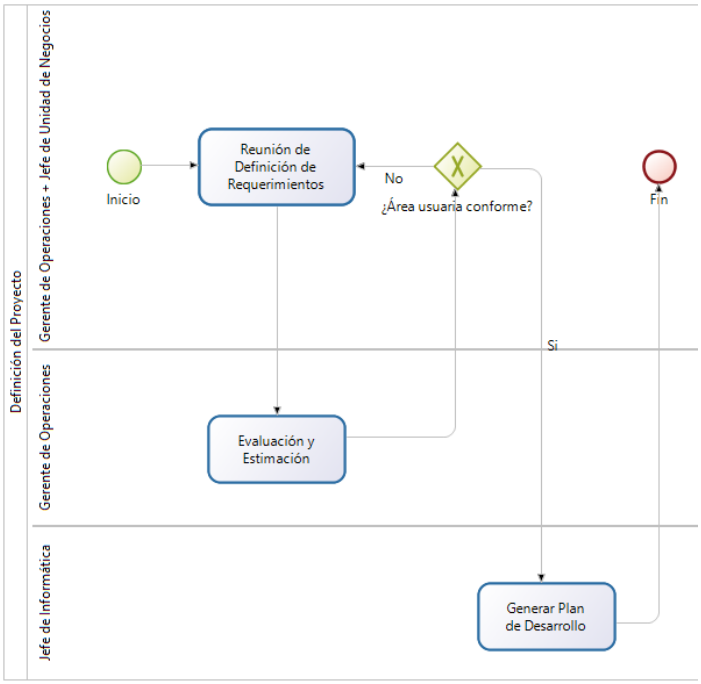


Figura 46: Diagrama Definición del Proyecto - ProcesosSoftware – BPMN Carril Separado

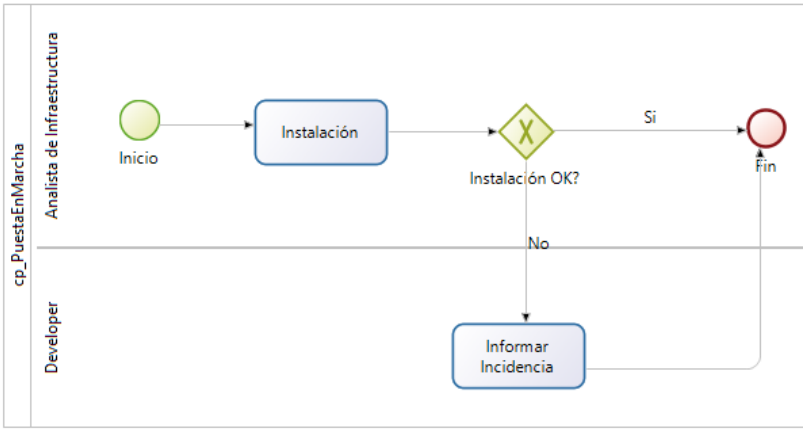


Figura 47: Diagrama cp_PuestaEnMarcha - ProcesosSoftware – BPMN Carril Separado

En el subproceso “Desarrollo” se observan dos carriles. El último corresponde a un rol compuesto que nace debido a la tarea colaborativa “Pruebas” (Figura 48). Cabe destacar que la tarea “Captura de Requerimientos” corresponde a una actividad (compuesta de tareas en SPEM), la cual al no tener un rol responsable es asignada al primer rol que aparece (junto con las compuertas y los nodos de inicio y fin. El subproceso que nace a partir de esta actividad demuestra el uso de compuertas paralelas (Figura 49).

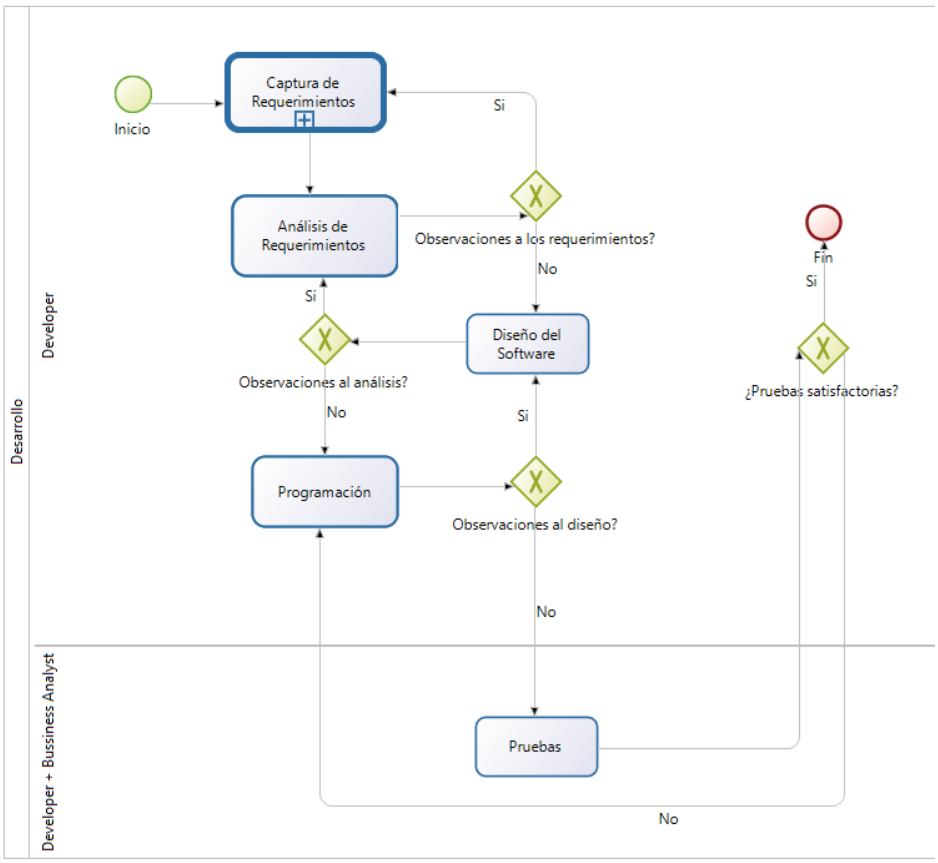


Figura 48: Diagrama Desarrollo - ProcesosSoftware – BPMN Carril Separado

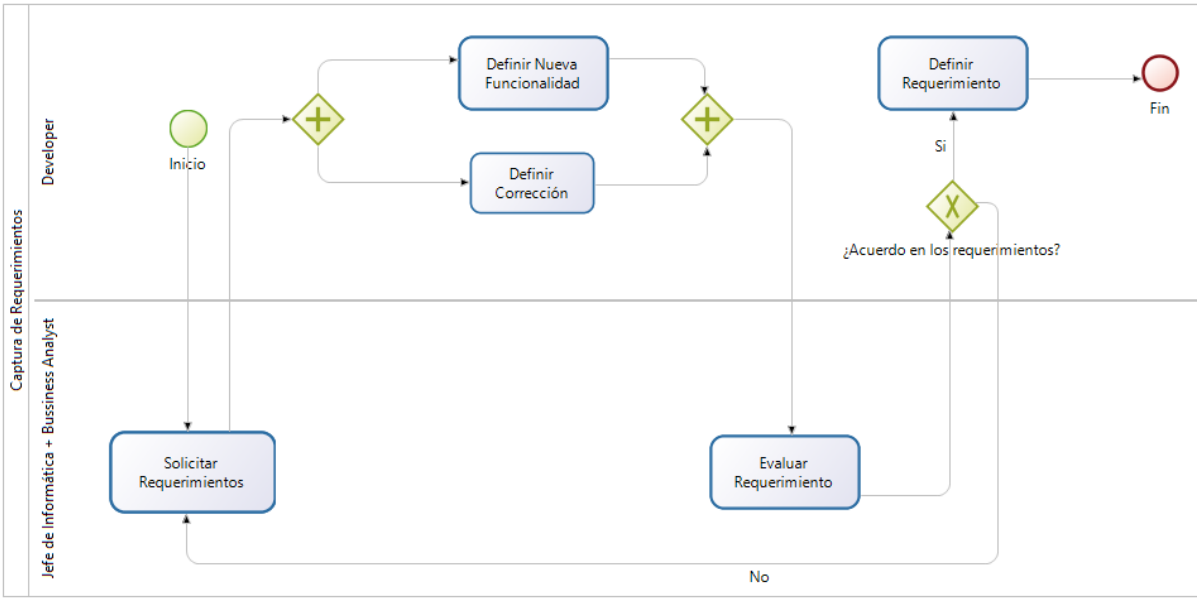


Figura 49: Diagrama Captura Requerimientos - Desarrollo - ProcesosSoftware – BPMN Carril Separado

4.2.2 Proceso Separado

Al aplicar esta transformación al Proceso de Incidencias, se puede apreciar la misma distribución que en el caso anterior para el proceso general (Figura 50).

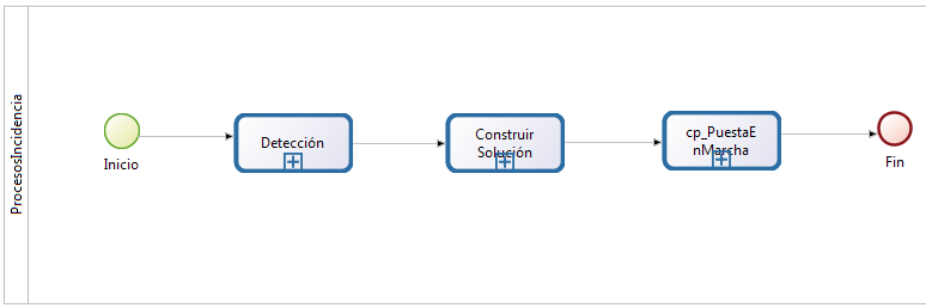


Figura 50: Diagrama General - ProcesosIncidencia – BPMN Proceso Separado

Con respecto al subproceso “Detección”, se aprecian tres carriles, uno para cada rol principal. Aquí se puede ver el primer “Meeting” generado por la transformación, correspondiente a la tarea colaborativa “Recepción del Incidente” (Figura 51).

Figura 51: Diagrama Detección - ProcesosIncidencia – BPMN Proceso Separado

En el subproceso “Construir Solución” se puede ver un carril único, correspondiente al rol principal de las tareas, incluyendo las tareas colaborativas. Es importante destacar que este diagrama “oculta” la participación del rol de “Analista de Negocios” en las tareas “Reproducción del Incidente” y “Pruebas”. Dicho rol debe aparecer en un subproceso con el prefijo “Meeting” (Figura 52).

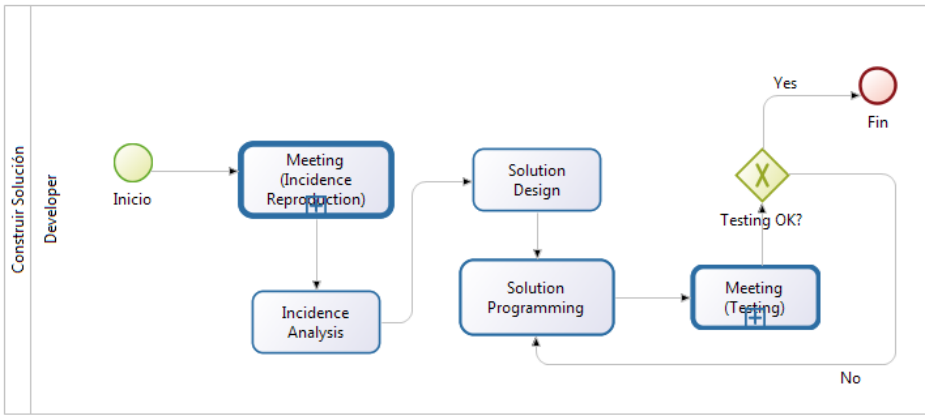


Figura 52: Diagrama Construir Solución - ProcesosIncidencia – BPMN Proceso Separado

El subproceso “cp_PuestaEnMarcha”, tiene la misma estructura de los casos anteriores, manteniendo la coherencia en la transformación (Figura 53).

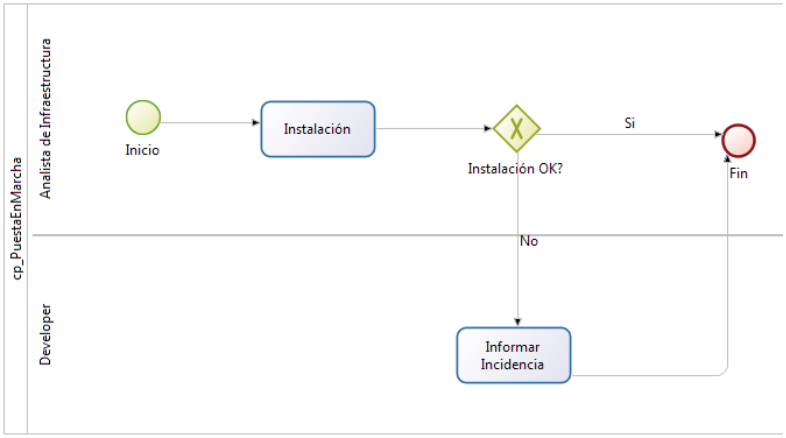


Figura 53: Diagrama cp_PuestaEnMarcha - ProcesosIncidencia – BPMN Proceso Separado

La gran diferencia de esta transformación radica en la construcción de un subproceso para cada tarea colaborativa. Cada uno de estos subprocesos posee un solo carril, el cual representa el rol compuesto que realiza la tarea colaborativa (incluyendo los roles que quedaron “ocultos” en los diagramas anteriores) y posee una sola tarea (con el

nombre de la tarea colaborativa) además de los respectivos nodos de inicio y fin (Figura 54).

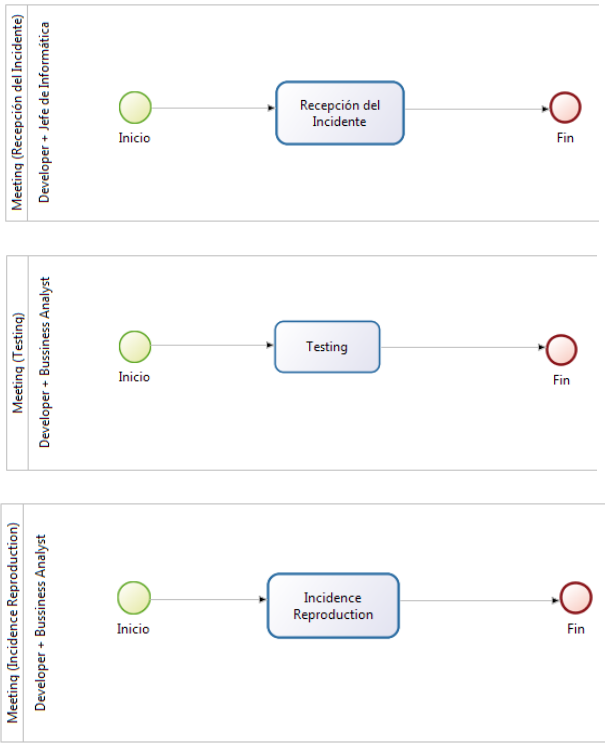


Figura 54: Diagrama Subprocesos Colaborativos - ProcesosIncidencia – BPMN Proceso Separado

Con respecto al Proceso de Desarrollo de Software, esta transformación genera un diagrama general que no posee mayores cambios respecto al definido en SPEM (Figura 55).

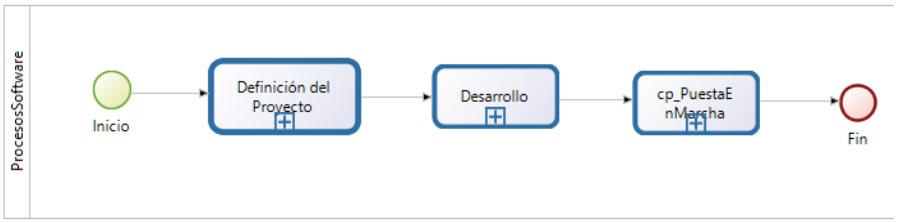


Figura 55: Diagrama General - ProcesosSoftware – BPMN Proceso Separado

En este caso, en el subproceso de “Definición del Proyecto” aparecen dos carriles, “ocultando” la participación del “Jefe de Unidad de Negocios” en la tarea “Reunión de Definición de Requerimientos”. Esta interacción deberá quedar explicita en el

subproceso generado por la transformación cuyo nombre será “Meeting” más el nombre de la tarea (Figura 56).

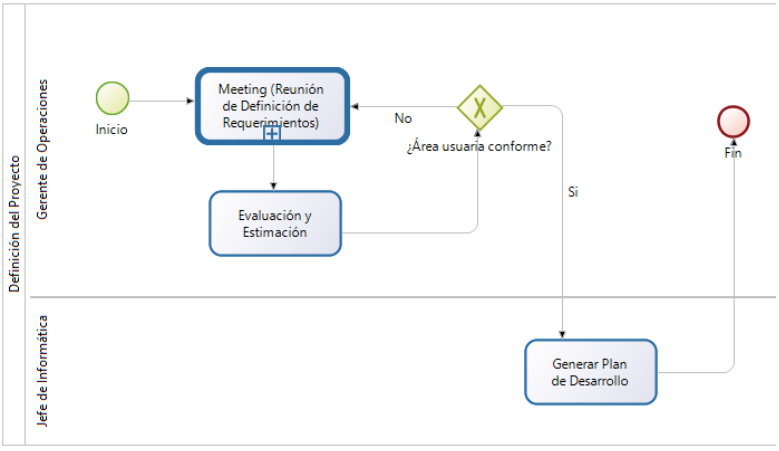


Figura 56: Diagrama Definición del Proyecto - ProcesosSoftware – BPMN Proceso Separado

Por parte del subproceso “Desarrollo”, aparece un carril y dos subprocesos. Uno de estos subprocesos corresponde a una actividad, y el otro a una tarea colaborativa. La tarea colaborativa debe traducirse en un subprocesos en donde aparezcan los involucrados. Este diagrama “oculta” la participación del rol “Analista de Negocios” (Figura 57).

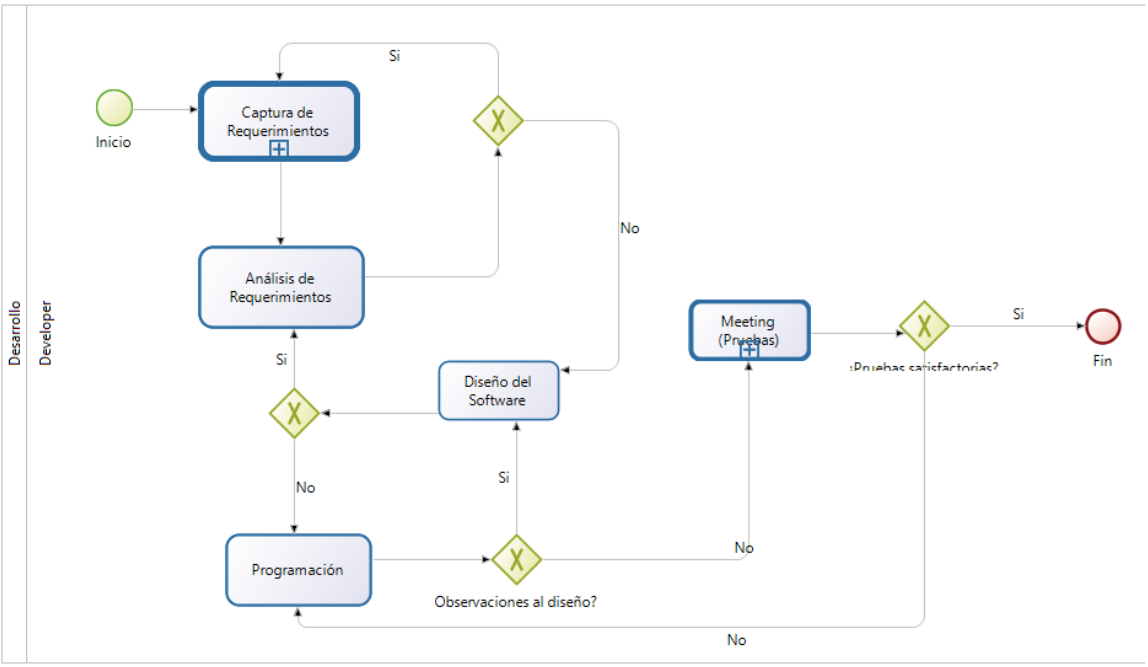


Figura 57: Diagrama Desarrollo - ProcesosSoftware – BPMN Proceso Separado

El subproceso “Captura de Requerimientos” nace de la definición de una actividad en SPEM en el proceso “Desarrollo”.

Este subproceso contiene dos tareas colaborativas las cuales deben traducirse, a su vez, en dos subprocesos. Este diagrama “oculta” la participación del rol “Analista de Negocios” en ambas tareas (Figura 58).

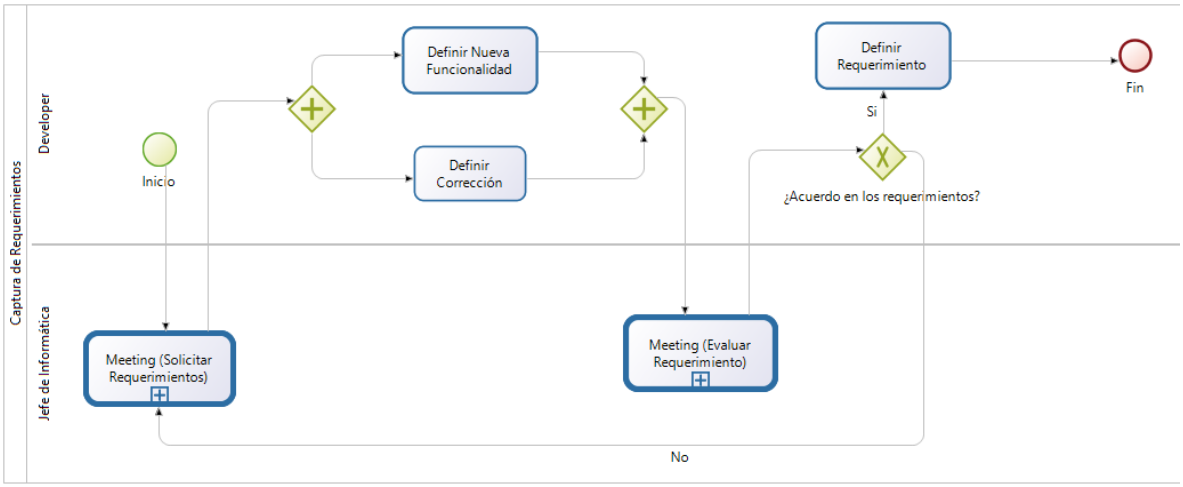


Figura 58: Diagrama Captura de Requerimientos - Desarrollo - ProcesosSoftware – BPMN Proceso Separado

El subproceso “cp_PuestaEnMarcha” se mantiene exactamente igual que en los casos anteriores (Figura 59).

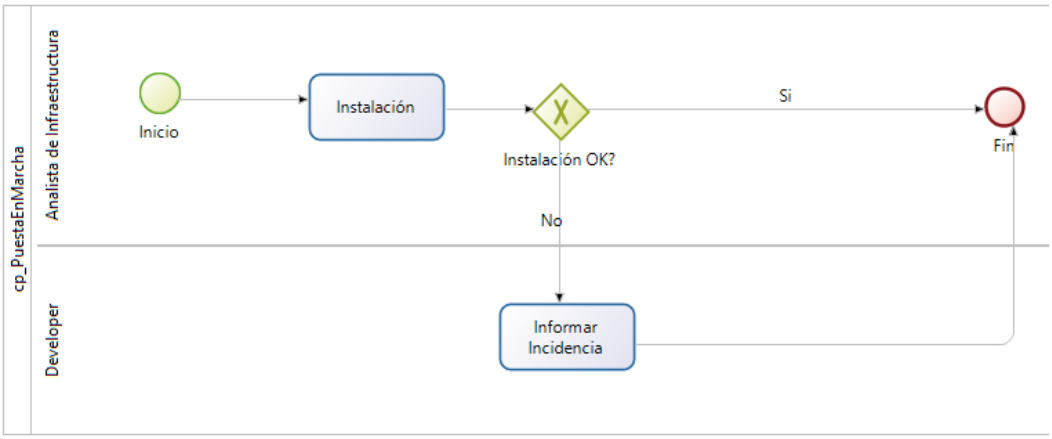


Figura 59: Diagrama cp_PuestaEn Marcha - ProcesosSoftware – BPMN Proceso Separado

Finalmente, se encuentran todos los subprocesos creados por la transformación que corresponden a las tareas colaborativas definidas en SPEM. Dichos subprocesos, contienen la totalidad de los roles responsables unidos en un solo carril (Figura 60).

Figura 60: Diagrama Subprocesos Colaborativos - ProcesosSoftware – BPMN Proceso Separado

4.3 Resultados del Caso de Estudio

Para determinar la equivalencia de los diagramas BPMN resultantes de la transformación estos han sido presentados en Librería Antártica, concretamente a la jefatura del Departamento de Sistemas y a la Gerencia de Logística de la cual dicho departamento depende.

La jefatura de Departamento de Sistemas, indicó que el modelo BPMN refleja la participación de todos los roles, mientras mantiene la secuencia de las tareas. Sin embargo, dicha jefatura pone énfasis en la pérdida de las plantillas (Guidelines) contenidas en SPEM, las cuales no se reflejan en el diagrama BPMN resultante. Con respecto a la representación de las tareas colaborativas, esta jefatura se inclina por el uso de carril separado, ya que le permite ver en un solo proceso todas las interacciones entre los roles responsables.

Por otro lado, la Gerencia de Operaciones, destacó la posibilidad de compartir los procesos del Departamento de Sistemas con los otros departamentos, para poder transparentar a los responsables y justificar los tiempos y coordinar esfuerzos. Con respecto a la representación de las tareas colaborativas, dicha gerencia prefiere el uso de proceso separado, ya que le permite tener una visión de alto nivel de las tareas y sus responsables.

Considerando las opiniones de los involucrados, se puede concluir que el flujo y representación de tareas colaborativas son equivalentes en el modelo original en SPEM y en el modelo BPMN resultante de la transformación. Sin embargo, y debido a que el objetivo de ambas notaciones es diferente, en el modelo BPMN resultante se pierden algunos elementos útiles; como por ejemplo, los Guidelines definidos en SPEM.

5 Conclusiones

5.1 De procesos SPEM a procesos BPMN

El presente trabajo de investigación, parte de la hipótesis de que es posible transformar un modelo de proceso especificado en SPEM a uno equivalente en BPMN. Para probar dicha hipótesis, se han definido reglas de transformación de SPEM a BPMN (incluyendo la información de roles incluso en tareas colaborativas), y se ha construido una transformación automática. Dicha transformación automática permite reducir la complejidad de aplicar las reglas de transformación de forma manual, además de mejorar la coherencia entre los modelos evitando errores humanos, permitiendo además experimentar con diversos casos de la vida real.

Si bien esta transformación había sido llevada a cabo anteriormente, los resultados que se habían obtenidos son parciales o no se encuentran disponibles. Además, en los casos en que se lograba una transformación, esta no incluía todos los elementos SPEM, así como tampoco información de roles, ni mucho menos una representación de tareas colaborativas.

Los experimentos realizados han permitido determinar que la transformación entre los metamodelos es posible. Sin embargo, debido a que los objetivos de cada metamodelo son diferentes, existen ciertas limitaciones:

- SPEM permite definir algunos elementos sin un responsable (nodo de inicio y fin, decisión, Fork, Join, Merge, Activity e Iteration). Esto presenta un problema en BPMN a la hora de coexistir con el resto de las tareas. Dado este inconveniente, se ha decidido que la transformación asigne estos elementos al primer rol que aparezca en el proceso.
- SPEM permite definir un elemento de tipo “Milestone” que no ha sido transformado a BPMN debido a que no se ha encontrado un equivalente desde la perspectiva de la definición de procesos.
- SPEM permite definir tareas sin responsables. Dado que estas tareas pueden interactuar con otras que si tengan un responsable definido, se ha decidido que las tareas sin responsable sean asignadas en el modelo BPMN resultante a un carril “Undefined”.
- SPEM permite asociar información adicional a cualquier elemento del modelo, además de definir entradas y salidas de las tareas del proceso mediante el uso de Guidances y WorkProducts respectivamente. Sin embargo, a pesar de que BPMN permite el uso de Objetos de Datos que permiten representar elementos adicionales a una tarea, no se ha considerado su inclusión en el modelo BPMN resultante, principalmente debido a la complejidad que esto agrega a la transformación.

Adicionalmente, parte de los experimentos realizados fueron tomados de procesos de software de la vida real. Estos experimentos han permitido verificar la equivalencia del proceso BPMN resultante, así como la utilidad de contar con una representación de los procesos del área sistemas que sean comprensibles por otras áreas, facilitando la comunicación, colaboración y coordinación.

Finalmente, a la luz de los experimentos realizados, se puede concluir que es posible transformar un modelo de proceso desde el metamodelo SPEM a otro equivalente en el metamodelo BPMN, lo cual válida la hipótesis de trabajo. Por otra parte, este trabajo ha entregado herramientas y alternativas para agregar la información de los roles e incluir tareas colaborativas en el modelo BPMN resultante.

5.2 Hacia un proceso ejecutable

Existen diversas herramientas BPMS que permiten ejecutar un proceso definido en BPMN. Estas herramientas toman la definición del proceso de negocio (como un modelo BPMN) y la definición técnica de la herramienta a generar (base de datos, formularios, conectores, etc.) para generar un software que automatiza el proceso.

En este contexto, la obtención de un modelo BPMN a partir de un modelo definido en SPEM representa un acercamiento a un la obtención de un proceso ejecutable desde SPEM. Sin embargo, la definición del ambiente técnico para la automatización del proceso aún está pendiente.

Este acercamiento a procesos ejecutables plantea también algunos desafíos a la transformación del modelo SPEM. Por ejemplo, como se dijo anteriormente, las decisiones en SPEM no tienen un responsable, mientras que en BPMN las decisiones corresponden típicamente a la evaluación de un estado o un dato ingresado en alguna tarea por algún rol responsable, por lo que cabría preguntarse, ¿Será necesario crear una tarea BPMN para el cambio o ingreso de estado antes de una decisión, aunque dicha tarea no exista en SPEM?, ¿A qué rol se deberá asignar la tarea y la toma de decisión?

5.3 Oportunidades de Investigación

Este trabajo de investigación y sus resultados abren nuevas oportunidades de investigación que permitirían potenciar aún más su alcance.

En primer lugar, y como se mencionó en la sección anterior, es de sumo interés la extensión de la transformación para incluir elementos técnicos que permitan obtener un proceso automático a partir de un modelo definido en SPEM.

Adicionalmente, la transformación propuesta utiliza los “Pools” de BPMN para representar un proceso o actividad, mientras que los “Lanes” o carriles fueron utilizados para representar roles internos. Sin embargo, en BPMN un “Pool” es una representación gráfica de un “Participant” en una colaboración, el cual puede ser un “PartnerEntity” (como una empresa u organización) o un “PartnerRole” (como un comprador, vendedor o fabricante). Dado lo anterior, surge la pregunta, ¿Qué ocurriría con el modelo resultante al modelar roles externos (como clientes o usuarios finales) en “Pools”? Lo más probable es que la implementación de la transformación aumente en gran medida su complejidad.

Por otro lado, la transformación automática no contiene información de los elementos gráficos, por lo que, dependiendo del editor BPMN utilizado, el orden de los elementos gráficos no es el mejor, como se puede apreciar en la Figura 61. Luego resulta interesante la aplicación de un algoritmo de ordenamiento de los elementos gráficos, que permita una visualización más fácil del modelo BPMN resultante.

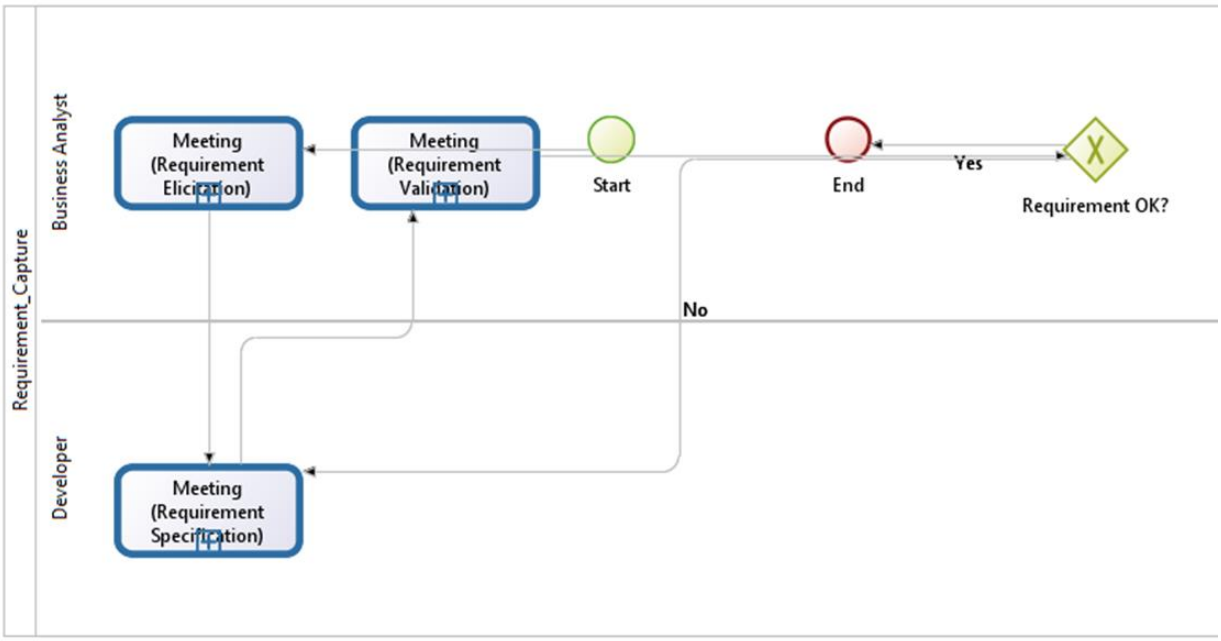


Figura 61: Diagrama sin orden de elementos gráficos en Bonita

Finalmente, la transformación automática sólo trabaja con los roles primarios definidos en SPEM. Sin embargo, SPEM permite definir roles primarios y adicionales, por lo que cabe preguntarse, ¿Cuál será el impacto de agregar dichos roles a la transformación? ¿Cuál será el impacto en el modelo BPMN resultante? Es muy probable que el modelo BPMN aumente en su complejidad, pero ¿Aumentará su capacidad para representar los roles dentro de una organización?

5.4 Artículo y Participación en Workshop

La presente investigación fue resumida y llevada a un artículo titulado “Transforming Multi-role Activities in Software Processes into Business Processes”.

Dicho artículo fue aceptado y presentado en el “1st Workshop on Resource Management in Business Processes”[12] en el marco de la conferencia “BPM 2016”[13] que

[12] https://ai.wu.ac.at/rema2016/
[13] http://bpm2016.uniriotec.br/

se realizó entre el 18 y el 22 de Septiembre de 2016 en Río de Janeiro, Brazil. Esta es la mayor y más importante conferencia internacional sobre modelamiento de procesos.

La buena recepción, por parte de participantes y organizadores del artículo basado en esta investigación permite confiar en la idoneidad y pertinencia de la investigación realizada.

Adicionalmente, esta presentación ha permitido obtener retroalimentación de otros investigadores del área BPMN, quienes han realizado diversos trabajos para determinar la utilidad de su uso para describir procesos de desarrollo de software.

Destaca la retroalimentación entregada por uno de los “Industry Track Speakers” de la conferencia, Marlon Dumas, profesor de Ingeniería de Software en la Universidad de Tartu, Estonia. Entre sus últimas publicaciones, se encuentra [22], en donde entrega directrices para el uso de BPMN en el modelado de procesos de desarrollo de software.

6 Bibliografía

[1] Object Management Group. 2008. Software & Systems Process Engineering Meta-Model Specification [en línea] < http://www.omg.org/spec/SPEM/2.0/ > [consulta: 15 de Septiembre de 2016]

[2] Menéndez V., Castellanos M. 2015. SPEM: Software Process Engineering Metamodel [en línea] < http://revistas.unla.edu.ar/software/article/view/672 > [consulta: 20 de Diciembre de 2016].

[3] Eclipse Foundation. 2008. Key Capabilities of the Unified Method Architecture (UMA). [en línea] <http://epf.eclipse.org/wikis/openupsp/base_concepts/guidances/concepts/introduction_to_uma,_94_eoO8LEdmKSqa_gSYthg.html> [consulta: 20 de Diciembre de 2016].

[4] Eclipse Foundation. 2008. Eclipse Process Framework Project [en línea] <https://projects.eclipse.org/projects/technology.epf> [consulta: 21 de Diciembre de 2016]

[5] IBM Corporation. 2005. Concepto: UMA contra el metamodelo RUP 2003 [en línea] <http://betaniatech.com/SmallProjects/core.base_concepts/guidances/concepts/uma_vs_rup_45521141.html> [consulta: 26 de Diciembre de 2017]

[6] La Rosa, V. 2013. EPF Composer: complemento del Proceso de Mejoras en la Universidad de Ciencias Informáticas [en línea] < https://www.redib.org/recursos/Record/oai_articulo983468-epf-composer-complemento-proceso-mejoras-universidad-ciencias-informaticas > [consulta: 03 de enero de 2017]

[7] Haumer, P. 2007. Eclipse Process Framework (EPF) Composer 1.0 Architecture Overview [en línea] <https://eclipse.org/epf/general/EPFComposerOverviewPart1.pdf> [consulta: 03 de Enero de 2017]

[8] Tuft B. 2010. Eclipse Process Framework (EPF) Composer: Installation, Introduction, Tutorial and Manual [en línea] <https://eclipse.org/epf/general/EPF_Installation_Tutorial_User_Manual.pdf> [consulta: 03 de Enero de 2017].

[9] White S., Miers D. 2009. BPMN – Guía de Referencia y Modelado. Future Strategies Inc.

[10] Hitpass, B. “et al”. 2014. BPMN 2.0 Manual de Referencia y Guía Práctica. Cuarta Edición. BPM Center.

[11] Bayard, O. 2011. INTRODUCCIÓN A BPMN [en línea] <http://bpmn-bayard.blogspot.cl/> [consulta: 05 de Enero de 2017].

[12] Ruiz, F. 2010. Procesos de Negocio y Desarrollo de SW [en línea] <http://alarcos.esi.uclm.es/per/fruiz/cur/santander/fruiz-pn.pdf> [consulta: 10 de Enero de 2017]

[13] Noguera, M. 2011. Introducción al Modelado de Procesos de Negocio. [en línea] <http://www.ugr.es/~mnoguera/collaborative_systems-business_processes_10-11.pdf> [consulta: 10 de Enero de 2017]

[14] Hevia, A. 2010. Ventajas del BPM [en línea] <http://pensandoensoa.com/2010/04/27/para-que-sirve-bpm/> [consulta: 10 de Enero de 2017]

[15] Robert Yin. 2009. Case Study Research: design and methods. Cuarta Edición, Sage Publications.
[16] Tuft, B. 2010. Eclipse Process Framework (EPF) Composer, Installation, Introduction, Tutorial and Manual [en línea] <https://eclipse.org/epf/general/EPF_Installation_Tutorial_User_Manual.pdf> [consulta: 15 de Enero de 2017].
[17] Combemale, B. “et all”. 2009. Towards a Rigorous use of SPEM [en línea] <https://hal.archives-ouvertes.fr/file/index/docid/369873/filename/iceis06-401-poster-CCCC.pdf> [consulta: 15 de Enero de 2017].
[18] Zorzan F. “et all”. 2009. Transformación de Procesos de Desarrollo de Software Tipo SPEM a Procesos Workflow. Una Propuesta de Caso de Estudio: SmallRUP [en línea] <http://sedici.unlp.edu.ar/bitstream/handle/10915/19731/Documento_completo.pdf?sequence=1> [consulta: 20 de Enero de 2017]
[19] Shapiro R. “et all”. 2012. BPMN 2.0 Handbook. Segunda Edición. Future Strategies.
[20] Perez. M. “et all”. 2010. Transformations from SPEM Work Sequences to BPMN Sequence Flows for the Automation of Software Development Process [en línea] <http://dl.acm.org/citation.cfm?id=1878797> [consulta: 03 de enero de 2017].
[21] Cervera M. “et al”. 2012. A Model-Driven Approach for the Design and Implementation of Software Development Methods. [en línea] <http://www.igi-global.com/article/model-driven-approach-design-implementation/70927> [consulta: 03 de Enero de 2017]
[22] Dumas M., Pfahl D. 2016. Modeling Software Processes Using BPMN: When and When Not? [en línea] <https://link.springer.com/chapter/10.1007/978-3-319-31545-4_9> [consulta: 20 de Septiembre de 2016]
[23] Bonitasoft Inc. 2016. Import A Diagram [en línea] <http://documentation.bonitasoft.com/5x/bos-58/process-design/working-bonita-studio/diagram/import-diagram> [consulta: 20 de Diciembre de 2016]
[24] Merelo, J. 2012. Generación de páginas Web usando XSLT y XML [en línea] <http://geneura.ugr.es/~jmerelo/XSLT/> [consulta: 15 de Diciembre de 2016]
[25] Sintes B. 2015. XSLT: Transformaciones XSL [en línea] <http://www.mclibre.org/consultar/xml/lecciones/xml_xslt.html> [consulta: 20 de Diciembre de 2016].
[26] Mens T. “et al”. 2006. A Taxonomy of Model Transformation [en línea] <https://www.researchgate.net/publication/30814644_04101_Discussion_--_A_Taxonomy_of_Model_Transformations> [consulta: 10 de Abril de 2017]
[27] Hafner M. “et all”. 2006. Towards a MOF/QVT-Based Domain Arquitecture for Model Driven Security [en línea] <https://link.springer.com/chapter/10.1007/11880240_20> [consulta: 10 de Abril de 2017]
[28] Dehayni M. “et all”. 2009. Some Model Transformation Approaches: a Qualitative Critical Review [en línea] <http://www.aensiweb.com/old/jasr/jasr/2009/1957-1965.pdf> [consulta: 10 de Abril de 2017]
[29] Viet Coung N., Qafmolla X. 2013. Towards Automated Model Driven Development with Model Transformation Methods and Testing [en línea] < http://search.proquest.com/openview/3b284c2c0c66c6e38845f1278580d69e/1?pq-origsite=gscholar&cbl=2027420 > [consulta: 10 de Abril de 2017]

[30] Portela, C. “et al”. 2012. A Comparative Analysis between BPMN and SPEM Modeling Standards in the Software Processes Context [en línea] <http://file.scirp.org/pdf/JSEA20120500004_49614105.pdf> [consulta: 12 de Marzo de 2017]
[31] Pelechano V. 2016. MOSkitt4ME [en línea] <https://tatami.dsic.upv.es/moskitt4me/> [consulta: 10 de Marzo de 2017]
[32] Cruz E., Bastarrica M. 2014. De Procesos SPEM a Procesos BPMN. Un enfoque basado en MDE [en línea] <http://dci.ufro.cl/fileadmin/Cibse2014/CIBSE2014-SET_041-052.pdf> [consulta: 05 de Febrero de 2017]
[33] Pérez J. 2007. Notaciones y lenguajes de procesos. Una visión global [en línea] <https://www.lsi.us.es/docs/doctorado/memorias/Perez,%20Juan%20D.pdf> [consulta: 08 de Febrero de 2017]