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

Reporte Técnico RT 09-23

Especificación de un modelo de proceso académico de desarrollo de software utilizando SPEM-EPF

Florencia Polcaro Líber Batalla

Sebastián Pizard

2009

Especificación de un modelo de proceso
Académico de desarrollo de software utilizando
SPEM-EPF

Florencia Polcado, Líber Batalla, Sebastián Pizard

ISSN 0797-6410
Reporte Técnico RT 09-23
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 2009

Especificación de un modelo de proceso académico de desarrollo de software utilizando SPEM-EPF

Florencia Polcaro, Líber Batalla, Sebastián Pizard

Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Herrera y Reissig 565
Montevideo
Uruguay

Marzo 2009

Resumen

En este informe se presenta el trabajo realizado en el estudio y aplicación de SPEM (Software Process Engineering Meta-Model) y EPF (Eclipse Process Framework Project) en el marco de un módulo de taller del año 2009. Se estudio el estado del arte en la especificación de modelos de proceso y se realizó una primera especificación del Modelo de Desarrollo MUM — Modelo Unificado y Medible; que corresponde a un modelo académico símil RUP, utilizado en la asignatura Proyecto de Ingeniería de Software.

Palabras Clave: SPEM, Modelado de Procesos, EPF, MUM, Proyecto de Ingeniería de Software, Meta Modelos, Procesos, RUP.

1 Introducción

Los objetivos principales del módulo de taller son:

- Estudiar SPEM [1] (*Software Process Engineering Meta-Model*) y evaluar su posible aplicación a modelos de procesos ya conocidos.
- Estudiar y utilizar EPF [2] (*Eclipse Process Framework Project*) para realizar el modelado del modelo de proceso MUM [3] (Modularizado Unificado y Medible).

El presente documento pretende cubrir los aspectos abarcados a lo largo del módulo de taller e introducir al lector no familiarizado con éstos, con el fin de que pueda comprender el trabajo realizado.

El contenido de la documentación se divide a grandes rasgos en cuatro partes.

La primera parte está destinada a dar una breve introducción a los modelos de procesos base considerados para este proyecto.

La segunda parte de la documentación está dedicada al estudio de SPEM. Se describen las características generales del Meta-Modelo y se da un enfoque práctico de cómo utilizarlo, quienes deben utilizarlo y en que escenarios es recomendable su utilización.

La tercera parte, incluye aspectos prácticos relacionados con el Editor EPF. Se comienza dando una descripción funcional de EPF con el nivel de detalle necesario para que un usuario sin experiencias previas con el *framework*, adquiera los conocimientos básicos para utilizarlo. Luego se presenta el caso de estudio realizado, mostrando los resultados del modelado de MUM en SPEM. Finalmente se incluye una sección práctica destinada a explicar cómo publicar un proyecto en el editor EPF.

La cuarta y última parte de la documentación, lista las conclusiones y trabajos futuros que pueden ser complementar el trabajo realizado.

2 Procesos y Modelos de Proceso

Un proyecto exitoso es aquel que satisface las expectativas en los siguientes campos: costos, cronograma, y calidad (considerando la funcionalidad u otras características como parte de la calidad). Consecuentemente, cuando se planifica o ejecuta un proyecto de software, las decisiones son tomadas con el fin de reducir los costos, el tiempo, o bien para mejorar la calidad. Los proyectos de software se basan en procesos para organizar la ejecución de las tareas necesarias que les permitan cumplir con las metas en los distintos frentes de acción (costos, cronograma y calidad).

La especificación de un proceso para un determinado proyecto, define las tareas que deben realizarse en el proyecto y su orden de ejecución. Aunque la especificación de un proceso sea distinta del proceso real, muchas veces se los considera como la misma cosa. Sin embargo, aunque se asume que no hay dificultad en que un proyecto siga un proceso especificado, en la práctica no es tan simple. A menudo, el proceso seguido por el proyecto difiere bastante a la especificación del proceso para dicho proyecto. Las razones de estas divergencias pueden ser varias, desde "pereza" hasta falta de percepción de la importancia de ajustarse al proceso. Asegurar que el proyecto sigue el proceso que le fue planificado a medida es una tarea importante para las organizaciones en el negocio de "ejecutar proyectos".

Un modelo de proceso especifica un proceso genérico. Usualmente es especificado como un conjunto de etapas en las cuales el proyecto debe ser dividido, el orden en el cual las etapas deben ser ejecutadas, y restricciones o condiciones sobre la ejecución de las etapas. La premisa básica detrás de un modelo de proceso es que en las situaciones en las cuales el modelo es aplicable, usar el modelo de proceso para definir el proceso del proyecto llevará a obtener un bajo costo, alta calidad, y reducción del tiempo del ciclo. En otras palabras, un proceso es un medio para alcanzar las metas: alta calidad, bajo costo, y reducción de tiempo de ciclo. Y un modelo de proceso provee guías genéricas para desarrollar un proceso para un proyecto.

El proceso de un proyecto puede utilizar algún modelo de proceso. Esto es que el proceso del proyecto contiene semejanzas con el modelo de proceso. Sin embargo, usar un modelo de proceso no es simplemente traducir las tareas del modelo de proceso a las tareas del proyecto. Típicamente, para alcanzar los objetivos del proyecto, el proyecto requerirá un proceso que sea de alguna manera diferente al modelo de proceso. Esto es que el proceso del proyecto es generalmente una versión adaptada de un modelo de proceso genérico. Cómo debe ser adaptado el modelo de proceso para un proceso particular, depende de las características del proyecto. En resumen, el proceso de un proyecto puede ser obtenido de un modelo de proceso, adaptando este último para que se ajuste a las necesidades del proyecto. Para las organizaciones que utilizan procesos estándar, la adaptación de los mismos a sus proyectos particulares es una tarea importante.

Cuando se ejecuta un proceso sobre un proyecto, se producen productos de software, construyendo así el software final. O sea, un proceso especifica los pasos a seguir, el proyecto ejecuta los pasos, y durante el curso de ejecución se obtienen los productos de software. Un proceso limita los grados de libertad de un proyecto especificando qué tipos de actividades deben emprenderse y en qué orden. Debe quedar claro que es el proceso el que conduce un proyecto e influencia las salidas esperadas del mismo. Debido a esto, el foco de la ingeniería de software recae fuertemente en el proceso.

Cómo ya se dijo, uno de los requerimientos de este trabajo era considerar MUM como modelo de proceso base para pasar a SPEM mediante la utilización de EPF. Dicho modelo de proceso está basado en RUP (*Rational Unified Process*). El RUP es un modelo de proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo de software. El objetivo es asegurar la producción de software de calidad superior que satisfaga las necesidades de los usuarios dentro de un cronograma y presupuesto predecible [4].

2.1 Modelo de Proceso MUM

MUM es un modelo de proceso que sigue la metodología iterativa e incremental.

Además, este modelo de proceso está ciertamente restringido en su uso ya que fue concebido bajo la hipótesis de un equipo de trabajo compuesto entre 10 y 14 personas que trabajaran juntos en el curso "Proyecto de Ingeniería de Software" por un período de 14 semanas.

3 SPEM

3.1 Propósito

El RUP, el MUM y otros modelos de procesos están basados en un conjunto de ideas y conceptos que no están definidos de manera explícita ya que no se especifica el meta-modelo utilizado. Así mismo, el formato en que están disponibles dichas metodologías son documentos de texto en lenguaje natural. Esto conlleva a que toda la gestión de la información contenida en el modelo (creación, revisión, modificación, etc.) tenga que ser manipulada manualmente.

Con el objetivo de evitar esta situación, el OMG (*Object Management Group*) desarrolló el estándar SPEM versión 2.0, que pretende ser el estándar industrial para la representación de modelos de procesos de ingeniería de sistemas.

A continuación se presentan las principales características y conceptos de SPEM 2.0.

3.2 Modelado en SPEM 2.0

SPEM 2.0 es un meta modelo para modelos de procesos de ingeniería de software e ingeniería de sistemas. Sirve para definir procesos de desarrollo de software y sistemas y sus componentes. Vale recalcar que no es un lenguaje de modelado de procesos en general, dado que está orientado a los procesos de software.

La idea central de SPEM 2.0 para representar procesos está basada en tres elementos básicos: rol, producto de trabajo y tarea. Las tareas representan el esfuerzo a hacer, los roles representan quien lo hace y los productos de trabajo representan las entradas que se utilizan en las tareas y las salidas que se producen. La idea central subyacente es que un modelo de proceso consiste básicamente, en decir quién (rol) realiza qué (tarea) para, a partir de unas entradas (productos de trabajo) obtener unas salidas (productos de trabajo).

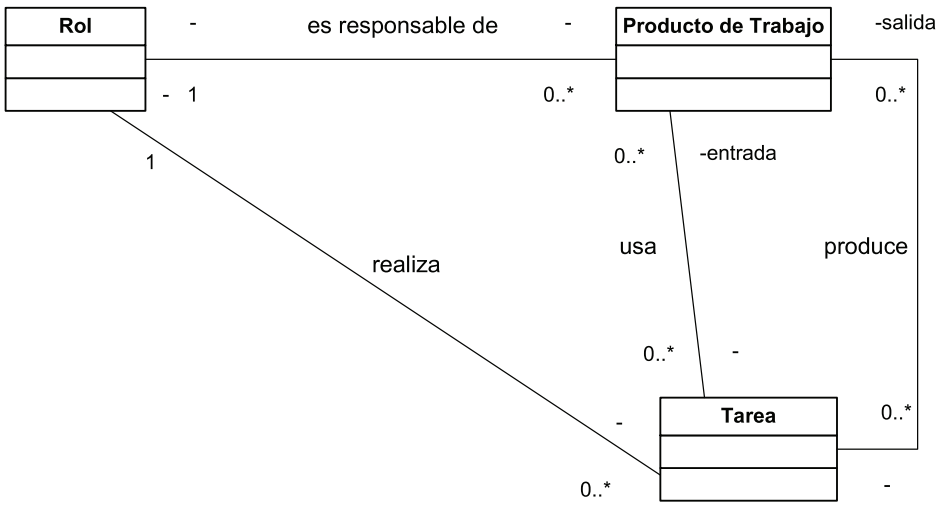

Figura 3.1 Idea central de procesos en SPEM 2.0.

¿Cuándo usar SPEM 2.0?

SPEM 2.0 es recomendado que sea aplicado en aquellas empresas que llevan a cabo proyectos de software y pueden enfrentar los problemas inherentes a esta disciplina. Entre dichos problemas, citamos:

- Los miembros de los equipos no cuentan con acceso fácil y centralizado a los procesos que necesitan.
- Diferentes desarrolladores manejan diferentes fuentes o versiones de la misma información.
- Es difícil integrar procesos e información que están en formatos diferentes.
- Cada libro, manual y herramienta utiliza un lenguaje y estilo diferente.
- Es difícil definir e implantar procesos de desarrollo que se puedan aplicar sistemáticamente y que se adapten a las necesidades.
- Es difícil reunir todo el *know how* de la organización: cultura, prácticas establecidas, procedimientos, requisitos de certificación, etc.

¿Quiénes deben usar SPEM 2.0?

Además de ser un meta-modelo para ingeniería de procesos, SPEM 2.0 también es un marco de trabajo conceptual que provee los conceptos necesarios para modelar, documentar, presentar, publicar, gestionar, intercambiar y realizar métodos y procesos de software. Por ello, está destinado a ingenieros de procesos, jefes de proyectos, gestores de proyectos y programas; que son responsables de mantener e implementar procesos para sus organizaciones o para proyectos concretos.

¿Cómo usar SPEM 2.0?

Al trabajar con SPEM 2.0 existen 4 escenarios fundamentales:

Crear un repositorio de "contenidos de métodos" reutilizables.

Siendo estos el conjunto de roles, tareas, productos de trabajo, métodos y procesos que definen la forma de trabajar de la organización. Esto ya es de gran valor para la organización ya que le permite tener en un formato estandarizado y unificado el conocimiento sobre los procesos.

Dar soporte al desarrollo, gestión y crecimiento de procesos de software.

SPEM 2.0 ayuda a los equipos de desarrollo a definir un proceso y a que puedan ser vistos como *workflows* o WBS, según interese en cada momento.

Establecer un marco de trabajo general de la organización a partir de los procesos y los elementos definidos anteriormente.

Para esto, SPEM 2.0 permite:

- Reutilización de procesos o patrones de procesos
- Variabilidad (procesos que incluyen partes alternativas configurables).
- Particularización (los usuarios pueden definir sus propias extensiones, omisiones y puntos de variabilidad sobre procesos estándares reutilizados).

En particular la variabilidad tiene fundamental aplicación para definir las extensiones al proyecto MUM (Extensiones: Orientada a Objetos, PSP, GX, Competisoft).

Generar plantillas para planes de proyecto concretos.

Para darles auténtico valor, las definiciones de los procesos deben ser desplegadas en formatos que permitan su realización automática (sistemas de gestión de proyectos y recursos, motores de flujos de trabajo, etc.).
Para ello, SPEM 2.0 incluye estructuras de definición de procesos que permiten expresar cómo un proceso será realizado de forma automática con estos sistemas. Ejemplos de ello

son las iteraciones (una o varias definiciones de trabajo serán repetidas varias veces en un proyecto) y las ocurrencias múltiples (varias instancias de una definición de trabajo pueden llevarse a cabo de forma paralela).

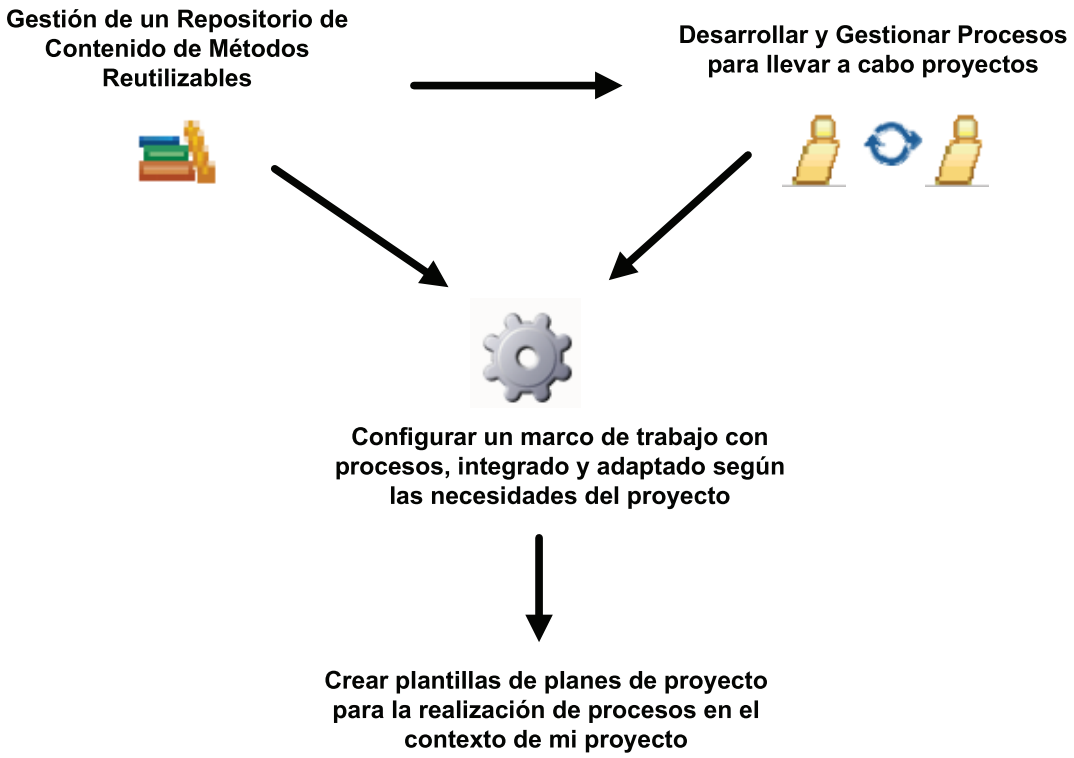

Figura 3.2 Marco de trabajo general con SPEM 2.0

3.3 Meta-modelo SPEM

En SPEM 2.0 se distinguen dos grupos de conceptos a la hora de implementar una metodología.

En primer lugar se definen los elementos del *Method Content* (el contenido del método) con *Content Elements* (elementos de contenido). Estos son los elementos primarios o constructores básicos.

En segundo lugar se combinan y reutilizan dichos elementos para obtener *Proceses* (Procesos).

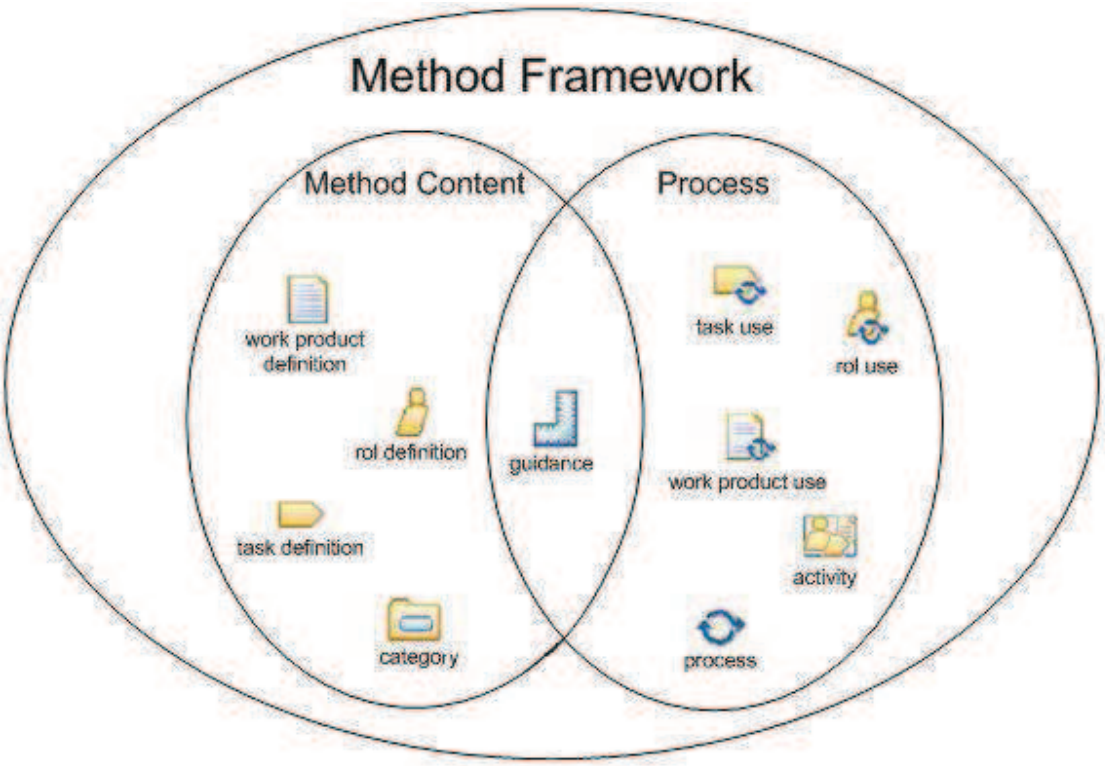

Figura 3.3 Aspectos principales para modelar SPEM 2.0.

3.3.1 Características Avanzadas de SPEM 2.0

Además de la separación clara entre la definición de contenidos de método y su aplicación en procesos, SPEM 2.0 cuenta con otras características avanzadas como ser:

Mantenimiento consistente de muchos procesos alternativos:

Permite tener diferentes variantes de procesos específicos, basados en los mismos contenidos de método y estructuras de procesos, pero aplicados con diferente detalle y escala.

Muchos ciclos de vida diferentes:

SPEM 2.0 permite trabajar con distintos tipo de ciclo de vida del software: cascada, iterativo, incremental, evolutivo, etc.

Variabilidad y extensibilidad:

SPEM 2.0 incluye un mecanismo de *plugins* de dos tipos: A) *Method plugins* para particularizar y adaptar contenidos de método sin modificar el original. B) *Process plugins* para procesos, pudiendo añadir o sustituir elementos de trabajo en el WBS sin afectar al original.

Patrones de proceso:

Son bloques reutilizables para crear nuevos procesos. Dicho trozo de proceso puede ser copiado y modificado, permitiendo individualizar el contenido del patrón o bien, puede ser aplicado por medio del mecanismo Actividad en Uso, que es una forma avanzada de reutilizar estructuras de proceso. Una Actividad en Uso define tipo de interrelaciones para que cuando el patrón esté siendo revisado o modificado, todos los cambios se reflejen automáticamente en todos los procesos en que se aplica el patrón.

Componentes de proceso:

Son piezas de proceso sustituibles y reutilizables basadas en los principios de encapsulación y caja negra. No se especifica la descripción del trabajo interna del componente. Esto permite manejar situaciones en que un proyecto requiere que ciertas partes del proceso no sean definidas hasta la ejecución o nunca, por ejemplo, en el caso de *outsourcing*.

4 *EPF composer*

EPFC (*Eclipse Process Framework Composer*), es una herramienta gratuita, desarrollada por el grupo ECLIPSE [4], que sirve para editar fragmentos de método, procesos o metodologías, y generar automáticamente la documentación adecuada en formato web. EPFC utiliza UMA (*Unified Method Architecture*), que a su vez está basada en SPEM 2.0.

En suma, EPFC es un editor de procesos SPEM 2.0 que incluye funcionalidades adicionales para publicar de forma automática sitios web.

4.1 Organización de un Repositorio en *EPF Composer*

Un repositorio o biblioteca de métodos y procesos en SPEM 2.0 (*Method Library*) es una colección de uno o más *plugins* y una o varias configuraciones (*Configuration*). Cada *plugin* incluye contenido de método (*Method Content*) y procesos (*Processes*), véase figura 3.3. A su vez, el contenido del método está formado por paquetes de contenido (*Content Package*), categorías estándar (*Standard Category*) y categorías personalizadas (*Custom Category*). El apartado de Procesos contiene patrones de proceso (*Capability Pattern*) y procesos para despliegue (*Delivery Process*). A continuación se muestra un ejemplo de esta estructura, disponible desde la vista "*Authoring*" de EPFC.

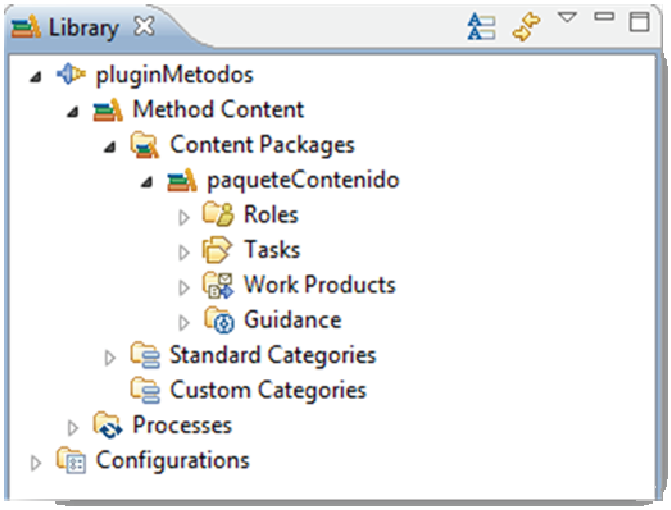

Figura 4.1 Árbol jerárquico de un proyecto en SPEM 2.0.

4.2 Contenido del Método

Esta sección está destinada a presentar los conceptos del Contenido del Método (*Method Content*) de SPEM 2.0.

El contenido del método puede ser organizado a voluntad del usuario mediante una jerarquía de paquetes de contenido (*Content Package*), cada uno de los cuales permite incluir roles, tareas, productos de trabajo y guías.

En la siguiente figura se muestra la jerarquía de conceptos empleados en contenido del método.

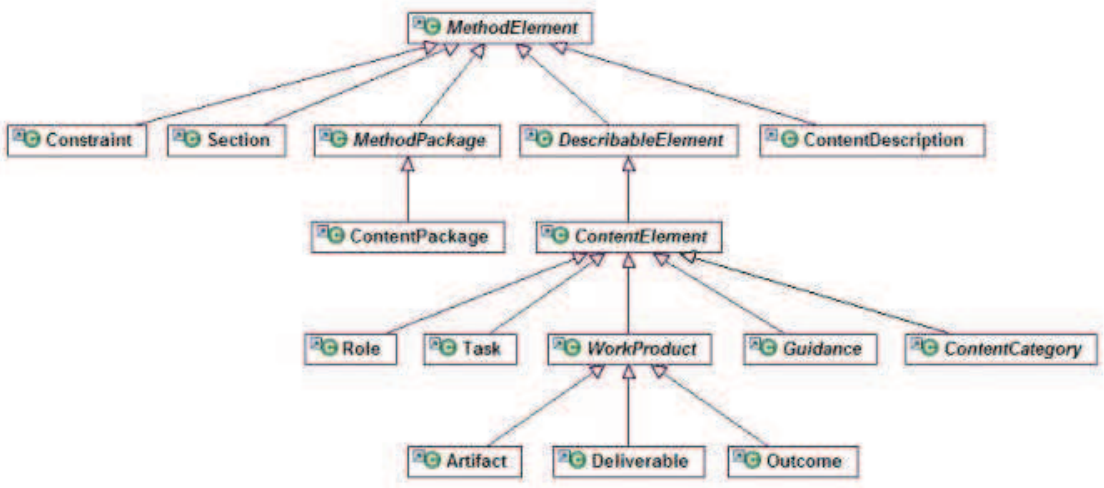

Figura 4.2 Jerarquía de conceptos del *Method Content*.

4.2.1 Elementos de Contenido

Los elementos de contenido (*content elements*) son los constructores básicos: tarea, rol, producto de trabajo, guía y categoría.

4.2.1.1 Tareas

Una tarea (*task definition*) describe una unidad de trabajo asignable y constituye una unidad atómica de trabajo para definir procesos. Su granularidad es de unas pocas horas a unos pocos días, afectando a unos pocos productos de trabajo y vinculando a unos pocos roles.

4.2.1.2 Roles

Un rol (*rol definition*) define un conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo. No se debe confundir roles con personas, ya que la vinculación entre personas y roles se realiza durante la planificación del proyecto y puede ocurrir que un individuo desempeñe varios roles, o que un rol sea desempeñado por varias personas. Un rol es un elemento de método usado en las definiciones de tareas para señalar quienes las realizan.

4.2.1.3 Productos de Trabajo

Un producto de trabajo (*work product definition*) es consumido, producido o modificado por tareas. Un producto de trabajo puede estar asociado con otros productos de trabajo mediante las siguientes asociaciones:

- Composición (*composition*), cuando las instancias de un producto de trabajo sirven para componer instancias de otro producto de trabajo.
- Agregación (*aggregation*), si un producto de trabajo está formado por agregación de otros.
- Es impactado por (*impact by*), cuando un producto de trabajo impacta en otro, es decir, si realizar cambios en el primero implican realizar cambios en el segundo.

Existen tres especializaciones de Productos de Trabajo:

- Artefacto (*artifact*): De naturaleza tangible (modelo, documento, código, archivos., etc.). Un artefacto puede estar conformado por otros artefactos más simples.
- Entregable (*deliverable*): Provee una descripción y definición para empaquetar otros productos de trabajo con fines de entrega a un cliente interno o externo. Representa una salida de un proceso que tiene valor para un usuario, cliente u otro participante. A su vez, puede estar asociado con componentes de entregable (*deliverable component*), que son los productos de trabajo, habitualmente artefactos, que lo forman.
- Resultado (*outcome*): Es un producto de trabajo de naturaleza intangible (resultado o estado), que no está formalmente definido.

4.2.1.4 Guías

Una guía o instrucción (*guidance*) es un elemento de método que provee información adicional relacionada con otros elementos. Por ejemplo: ayuda o información sobre cómo trabajo un rol, cómo crear un producto de trabajo, cómo usar una herramienta o cómo realizar una tarea.

4.2.1.5 Categorías

Una categoría (*category*) es un elemento de contenido, o de proceso, usado para clasificar o agrupar dichos elementos en base a los criterios de que desee el ingeniero de procesos. Una categoría puede tener cero o varias subcategorías. Esto permite establecer cualquier tipo de jerarquía de agrupamiento de elementos. SPEM 2.0 distingue dos clases de categorías:

- Estándar (*standard category*): Vienen predefinidas en SPEM 2.0 y se explican a continuación.
- Personalizada (*custom category*): Sirven para que el ingeniero de procesos pueda definir otras categorías nuevas.

En SPEM 2.0 se incluyen 5 tipos predefinidos de categorías:

Conjunto de roles (rol set): Sirven para agrupar roles con características en común.

Disciplina (discipline): Una disciplina es una colección de tareas que están relacionadas con un área principal de esfuerzo dentro de un proyecto. En la perspectiva tradicional suelen ser: requisitos, análisis, diseño, etc.

Dominio (domain): Permiten establecer una jerarquía de dominios, para clasificar productos de trabajo, con tantos niveles como se desee.

Herramienta (tool): Permite asociar herramientas con tareas, roles o productos de trabajo.

Clase de producto de trabajo (work product kind): Se incluye por compatibilidad con la versión 1 de SPEM. Se distingue de Dominio en que un producto de trabajo puede pertenecer a varias clases de producto de trabajo distintas. Ejemplo: El mismo artefacto puede incluirse dentro de "Documento de Análisis" y dentro de "Producto de Software".

5 Caso de Estudio: Pasaje de MUM a SPEM 2.0

En esta sección se detallan las tareas realizadas para el pasaje del modelo MUM a SPEM 2.0 utilizando el editor de procesos *EPF composer*.

5.1 Modelado con *EPF composer*

Una vez comprendidos los conceptos anteriormente descriptos sobre la herramienta *EPF composer*, se debió establecer el mapeo de conceptos existentes en el modelo MUM con los descriptores existentes en la herramienta.

El espacio de trabajo dentro de la aplicación, está dividido en "Perspectivas" (estructura usual en las aplicaciones basadas en Eclipse) las cuales simulan vistas del proyecto asociadas a un concepto o dimensión específica del mismo.

En el caso de *EPF composer*, se cuenta con dos perspectivas principales: *Authoring* y *Browsing*, las cuales permiten alternar entre la edición del proceso que se está desarrollando y la revisión del mismo a modo de vista preliminar. Esto permite visualizar interactivamente las modificaciones realizadas.

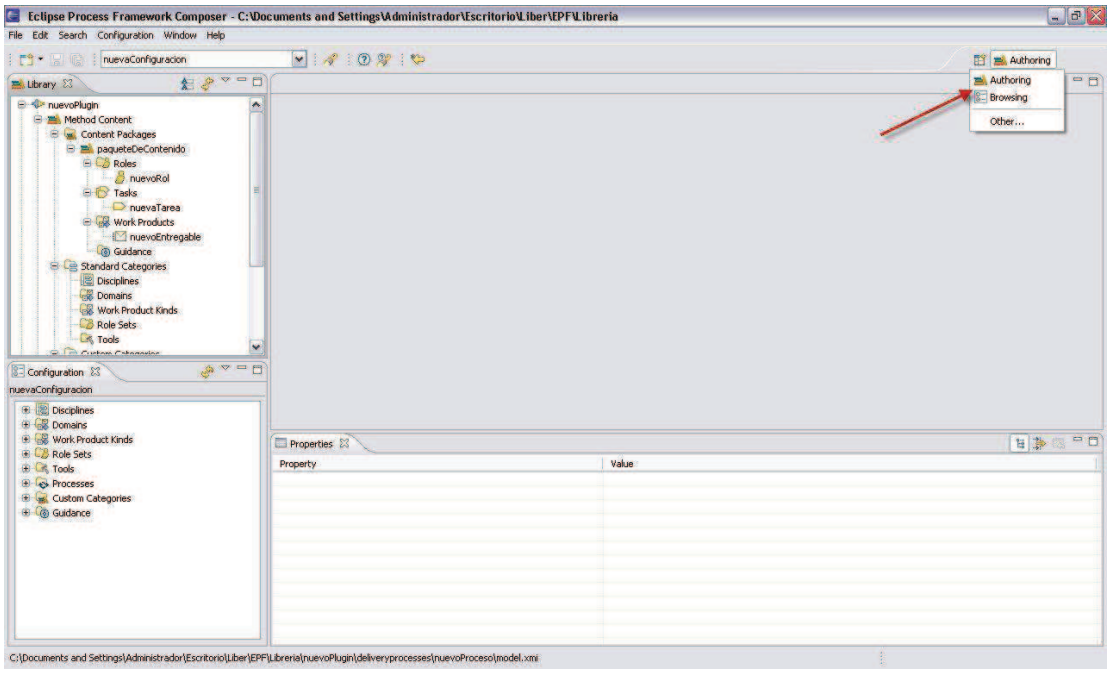

Figura 5.1 Espacio de trabajo indicando acceso a perspectivas.

Cada perspectiva se organiza mediante *Views* o vistas que son porciones de la pantalla para distintos usos.

La perspectiva *Authoring* contiene por defecto las vistas *Library*, *Configuration*, y *Properties* además de las ventanas de edición.

Mediante la vista *Library* se puede navegar en la estructura del proceso y desde ahí, iniciar la creación de contenido mediante un menú contextual, accesible mediante el ratón.

La estructura que se visualiza en esta vista puede no coincidir con la forma de almacenamiento del proyecto, por lo que no deberá confundirse con un explorador de archivos.

Se cuenta también con una perspectiva *Resource* (común a todos los entornos basados en la plataforma Eclipse) que permite ver el proyecto tal cual está almacenado en el disco duro imitando un explorador de archivos.

Para acceder a la misma, debemos realizar la apertura de la misma en la esquina superior derecha del espacio de trabajo (como se muestra en la figura).

Allí, deberemos elegir la perspectiva *Resource* de entre las disponibles.

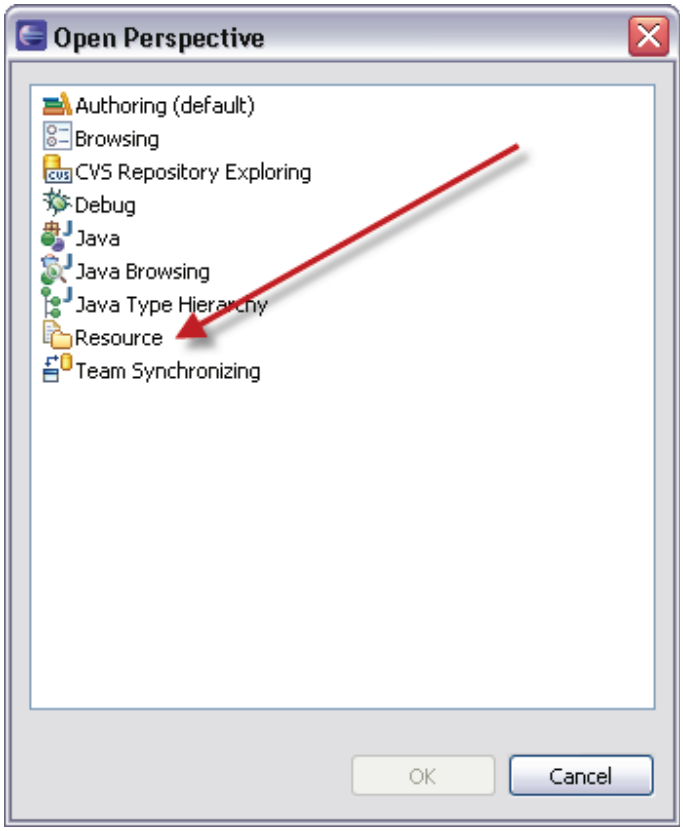

Figura 5.2 Selección de perspectivas.

Una vez allí, realizando la apertura de la vista *Navigator* (desde el menú de *Window*), observaremos la estructura física del proyecto.

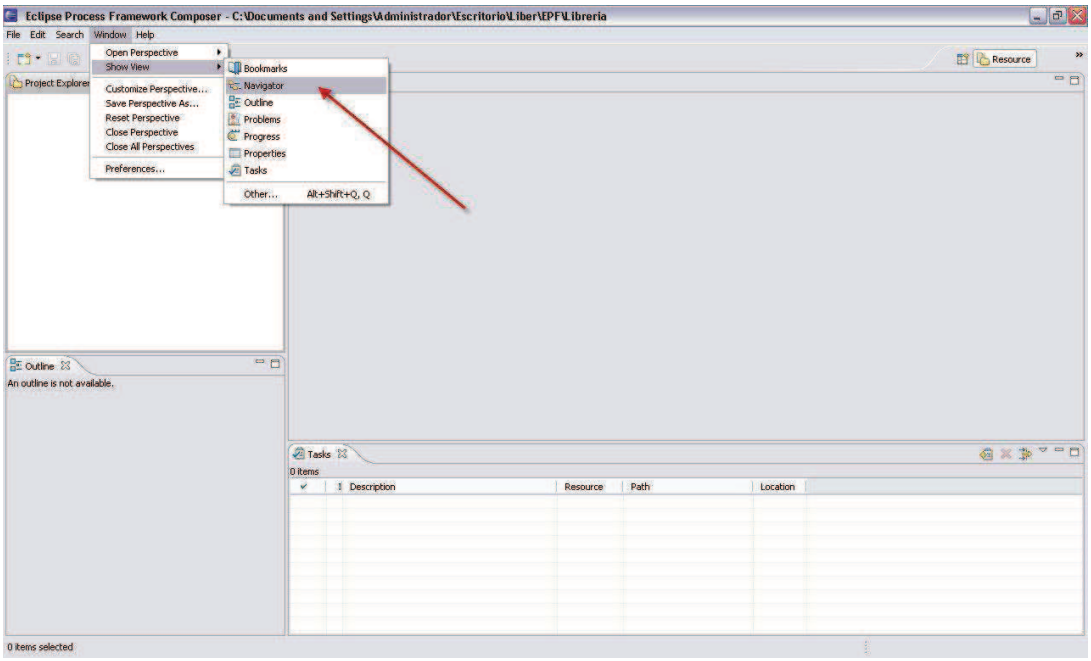

Figura 5.3 Menú de *Window* para la gestión de vistas de la perspectiva.

Como puede observarse en la imagen, la estructura no dista de la usual en proyectos Eclipse, la cual organiza los archivos en carpetas para cada *Configuration/Plugin*.

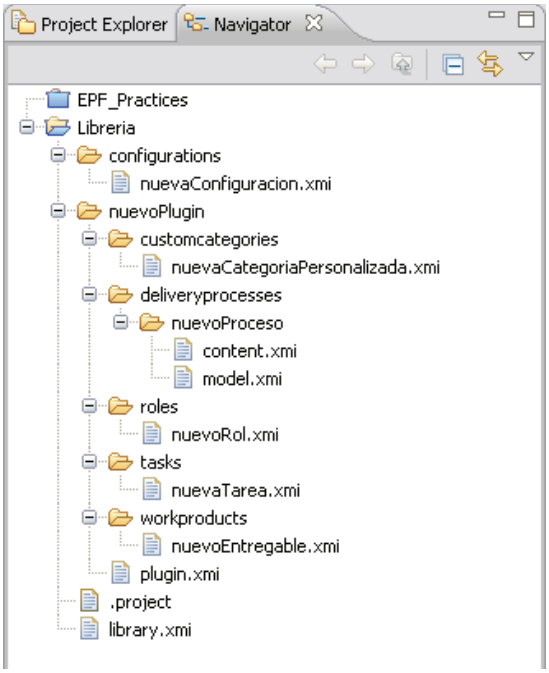

Figura 5.4 Vista *Navigator* para la gestión de los archivos de proyecto.

El software básicamente consiste en una interfaz de usuario para la gestión de contenidos (basada en *plugins* para la plataforma Eclipse) que genera páginas web tanto dinámicas como estáticas mediante meta-data almacenada en formato XML (de extensión XMI).

La edición de la información de los contenidos del proceso se edita también mediante páginas web que son mostradas en un navegador interno y son servidas mediante un servidor web interno al entorno.

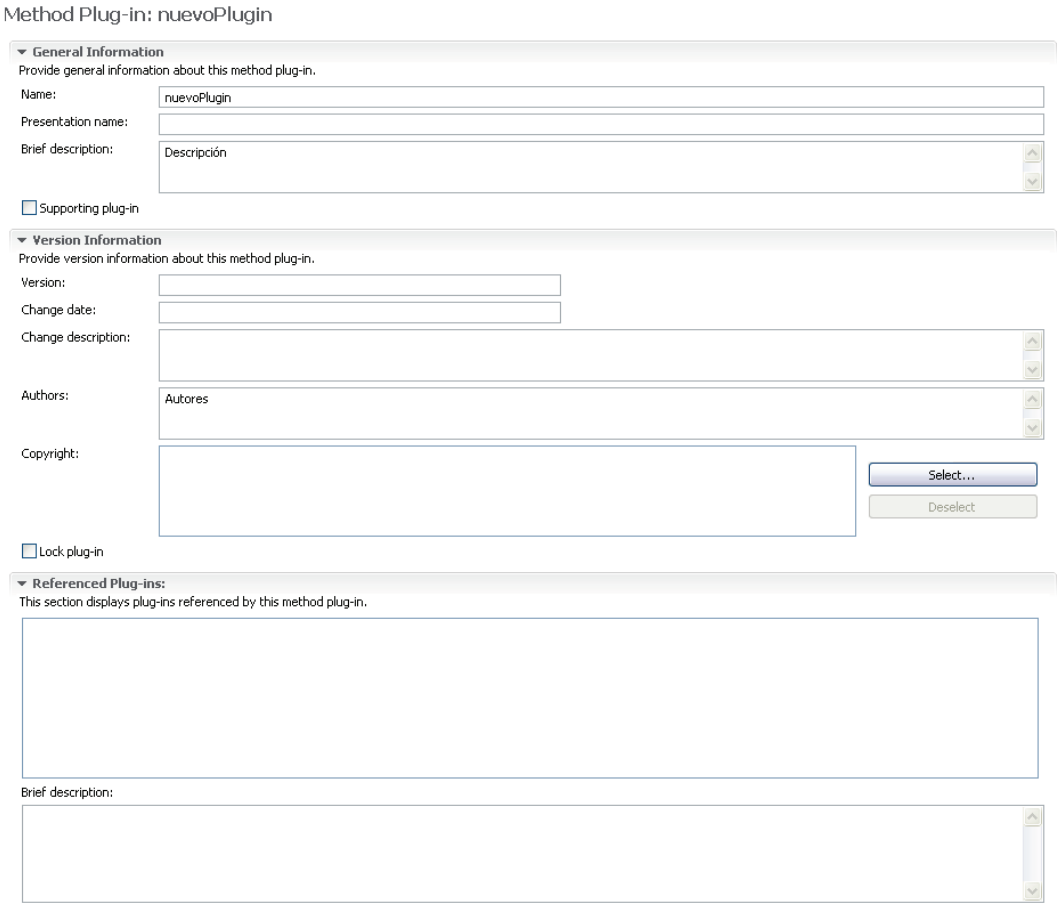

Figura 5.5 Ejemplo de formulario de edición de contenido (*Method Plug-in*).

Al momento de editar cualquier ítem de contenido, se estará ingresando información en una página web que se almacenará en archivos XML en la carpeta del proyecto. Esta información luego servirá para la generación de un sitio web con el contenido final (para los usuarios finales).

Es posible, en la mayoría de los formularios de edición de contenido, la previsualización de la página web a generar en el sitio web final. Dicha previsualización se localiza en la pestaña *Preview* debajo de cada formulario.

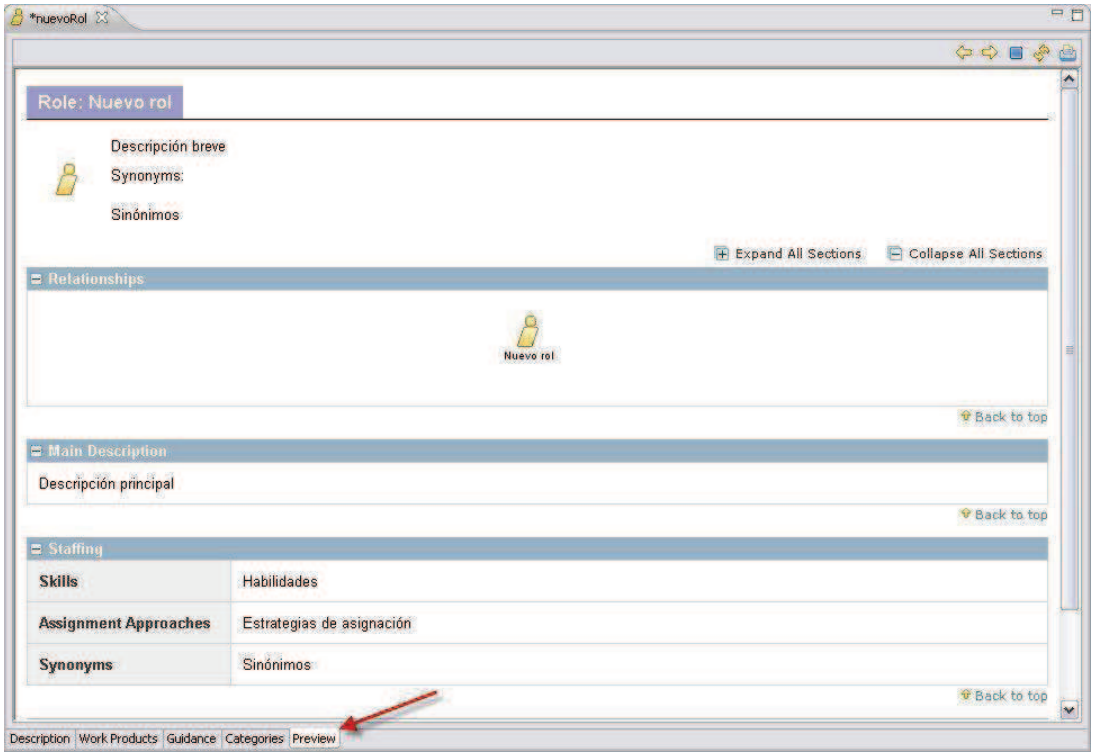

Figura 5.6 Ejemplo de previsualización de contenido.

Para comenzar a crear contenido, se deberá crear una *Method library* mediante el menú de *File*. De esta forma, se estará generando el repositorio de contenido de proceso.

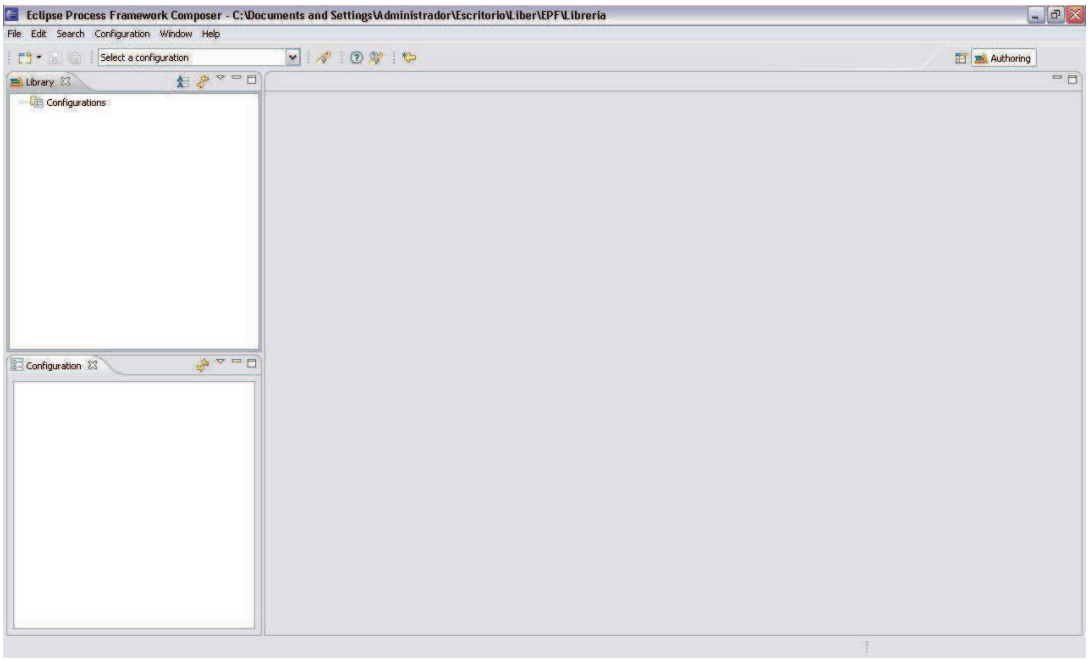

Figura 5.7 Espacio de trabajo al crear una nueva *Method library*.

Una vez creada la *Method Library,* se deben generar *Method Plugins* en los que se alojará el contenido reutilizable de procesos. Estos *Method Plugins* se crean a también mediante el menú *File* o con el menú contextual a la vista *Library*.

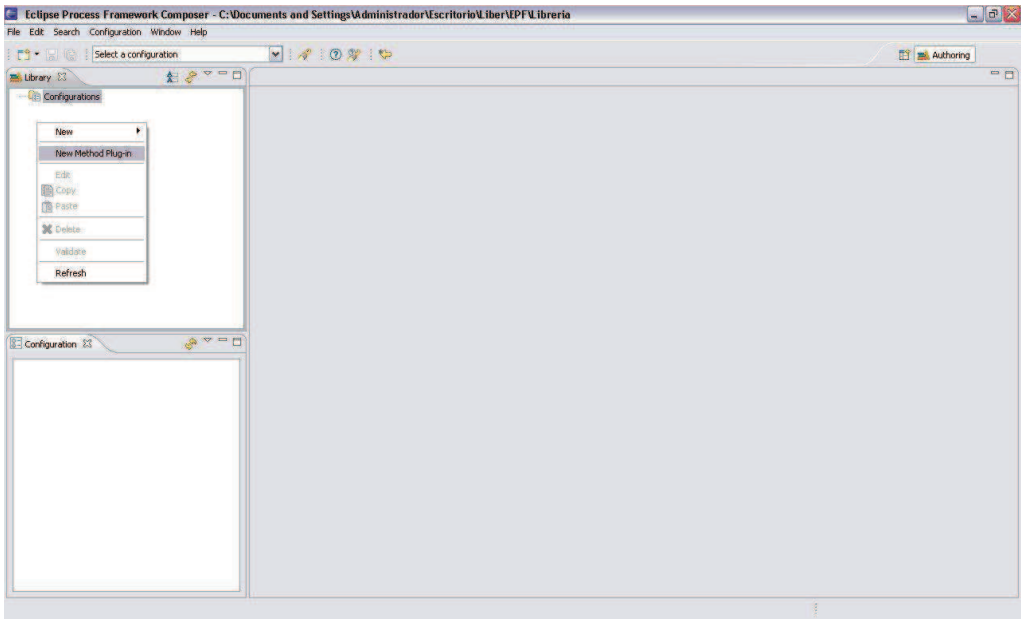

Figura 5.8 Menú contextual para la creación de un nuevo *Method Plug-in*.

Al crear un *Method Plugin* se replica una estructura para la organización del contenido (como se vio en el capítulo anterior) que abarca los tipos de ítems que se podrán crear.

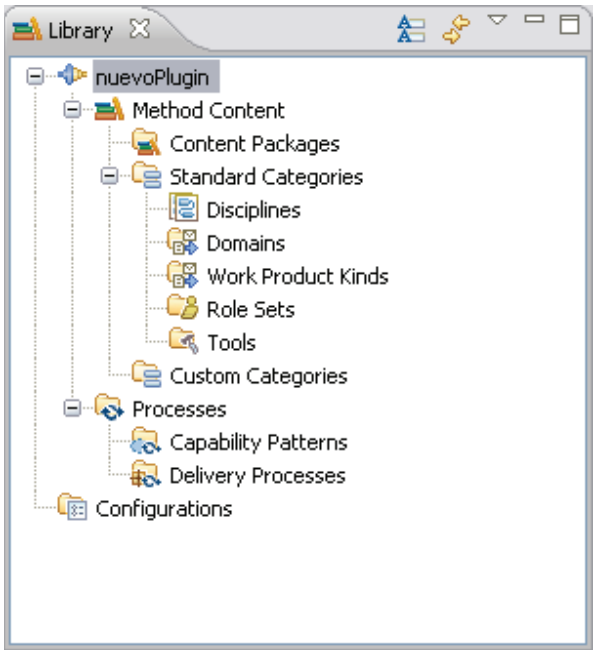

Figura 5.9 Vista de *Library* para la creación de contenido.

Cada ítem que vaya siendo creado se referenciará (cuando sea posible) con otros ítems en su ventana de edición. Por ejemplo, se podrá asociar una Tarea a una Dimensión.

Dentro de la estructura de cada *Method Plugin* se podrá crear contenido categorizado como de proceso *Processes* y de contenido *Content*.

Dentro de los contenidos de método, se puede subdividir los ítems en paquetes reutilizables mediante los *Content Packages*. Al crear *Content Packages* se describe cada ítem individual, que se vinculará luego en el proceso.

Existen categorías conceptuales dentro de las cuales se puede ubicar cada ítem: *Disciplines*, *Domains*, *Work Product Kinds*, *Role Sets*, *Tools*. Estas, son provistas por defecto como forma de distinguir tipos de contenido.

A su vez, la categorización de cada concepto es extensible mediante la posibilidad de creación de categorías personalizadas (*Custom Categories*) dentro de las cuales se podrán agrupar ítems de cualquier índole con un significado común personalizado.

Por defecto, un *Content Package* subdivide los contenidos en *Roles*, *Tasks*, *Work Products* y *Guidances*. Dentro de cada una existen subtipos que expresan detalles conceptuales. Por ejemplo, un *Work Product* puede ser del tipo *Deliverable*, *Outcome* o *Artifact* (las diferencias entre cada uno pueden ser revisadas en la especificación de SPEM 2.0).

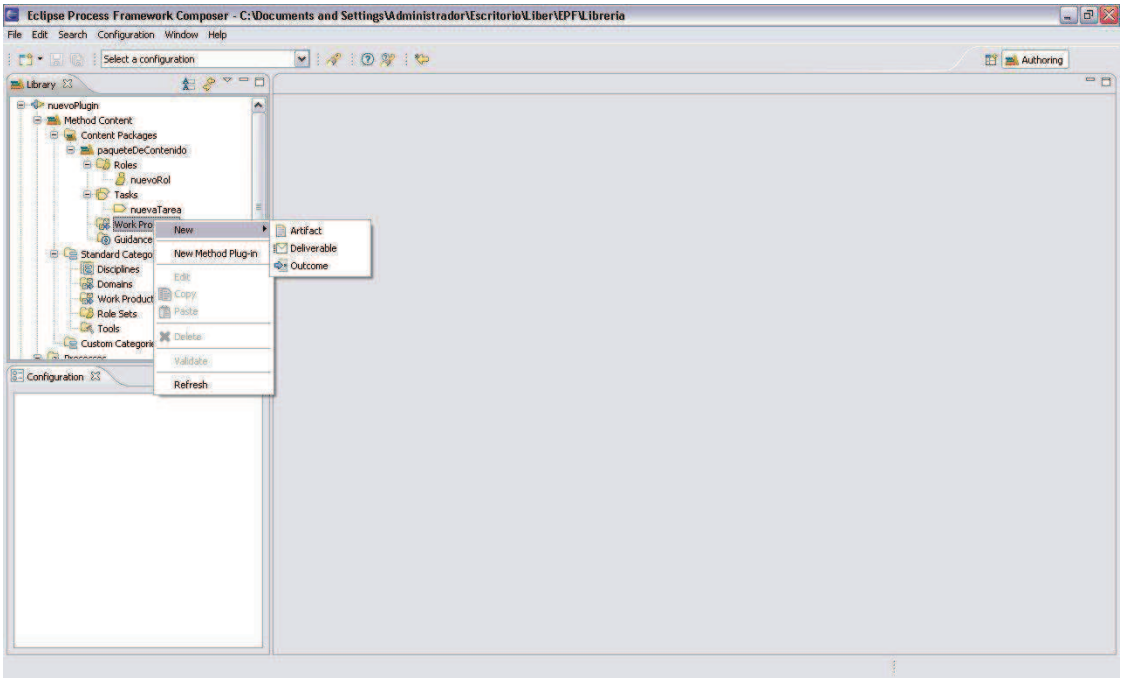

Figura 5.10 Creación de nuevo *Work Product*, mostrando los diferentes tipos disponibles.

Internamente, el almacenamiento se basa en un archivo XML que contiene los datos de cabecera de cada ítem y sus relaciones establecidas y además, para las propiedades extendidas (por ejemplo descripciones altamente personalizables como texto enriquecido, etc.) se crean archivos XML por separado que el editor mantiene asociados.

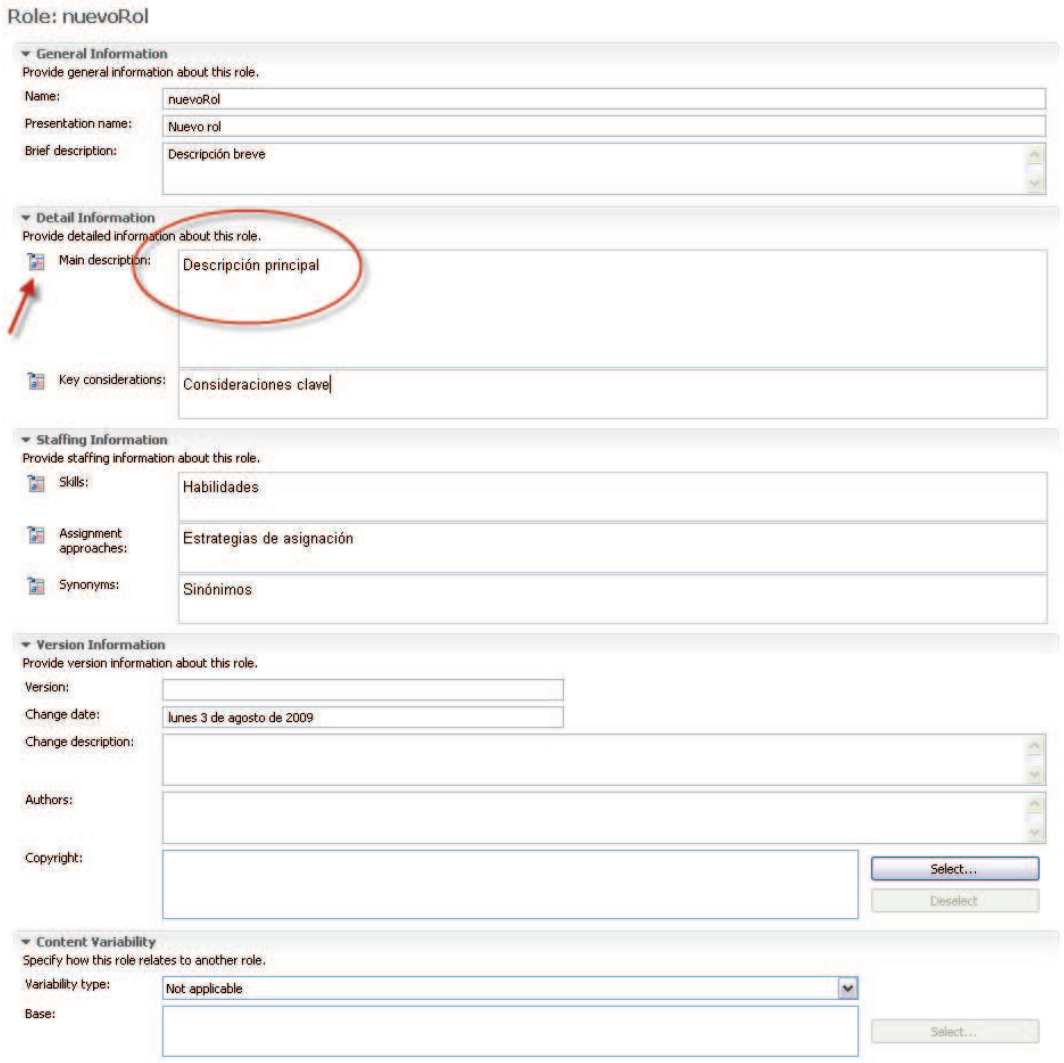

Figura 5.11 Ejemplo de formulario con campos personalizables de texto enriquecido.

Es importante tener esto en cuenta a la hora de portar el proyecto, ya que los archivos XML extras, son creados a demanda (sólo si son actualizados los campos personalizables). Estos campos, se identifican mediante el ícono de edición de texto enriquecido indicado en la figura.

Se recomienda que los campos personalizables sean editados tan pronto se cree cada ítem que los contenga. Esto generará la estructura de carpetas definitiva del proyecto Eclipse subyacente, lo que hará más simple su administración.

Para la creación del contenido, fue utilizada la nomenclatura *Camel Case* (para favorecer la portabilidad de los archivos fuente) ya que al agregar caracteres especiales (espacios, tildes, etc.) podría volver el proyecto incompatible de una plataforma a otra.

Cada ítem cuenta con un campo *Name* y un campo *Presentation name*, los cuales permiten disociar la identificación del ítem con su presentación, lo cual es útil para favorecer la portabilidad antemencionada.

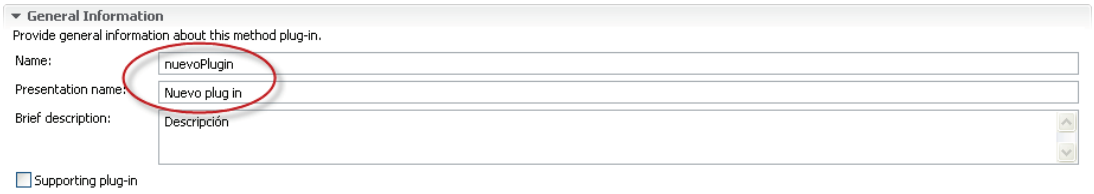

Figura 5.12 Ejemplo de campos *Name* y *Presentation name*.

Una vez creados los contenidos individuales y sus asociaciones, se podrán describir los procesos que los utilizan.

Hemos utilizado la categoría *Delivery Process* ya que el MUM se encuentra entre los procesos de desarrollo y esta refiere a los procesos específicos.

La opción *Capability Patterns* permite crear porciones reutilizables de procesos para la resolución de problemas comunes.

La edición de un proceso incluye una sección de descripción del mismo, un modelado en forma de WBS de las actividades a realizar, una vista del equipo de trabajo por etapa, y una vista de los productos de trabajo y otra consolidada.

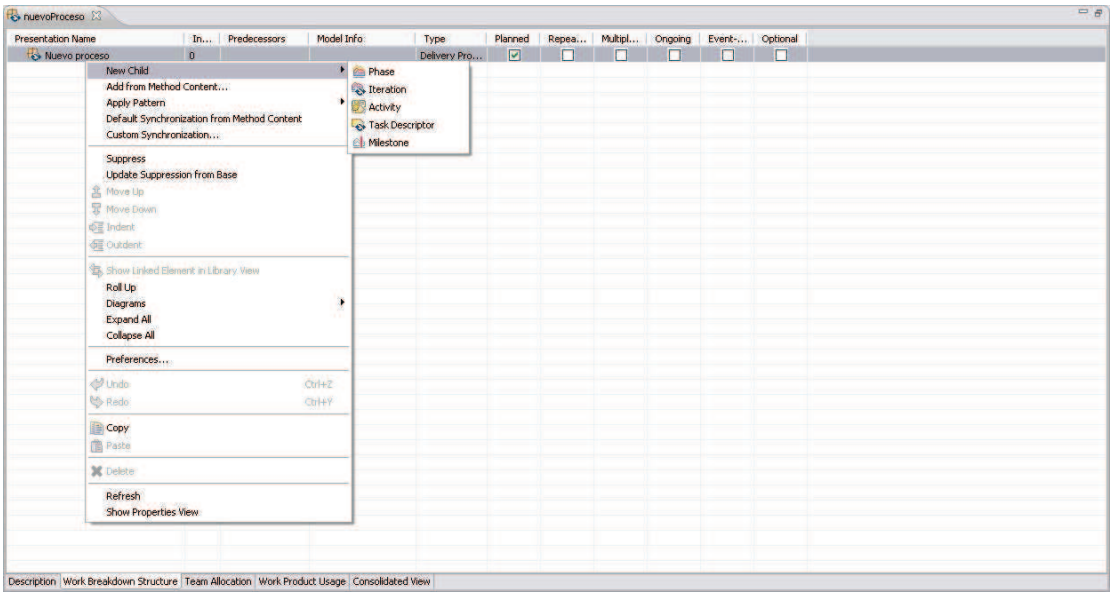

Figura 5.13 Ejemplo de estructura de proceso en formato *WBS*.

Cabe destacar que todas estas vistas se encuentran relacionadas, por lo que modificaciones en una se ven reflejadas en las demás.

5.2 Publicación del Proyecto

Una vez creados los ítems individuales y su utilización en forma de proceso, se podrán crear configuraciones (*Configurations*) que podrán incluir un subconjunto de los contenidos reutilizables de uno o varios *Method Plugins*.

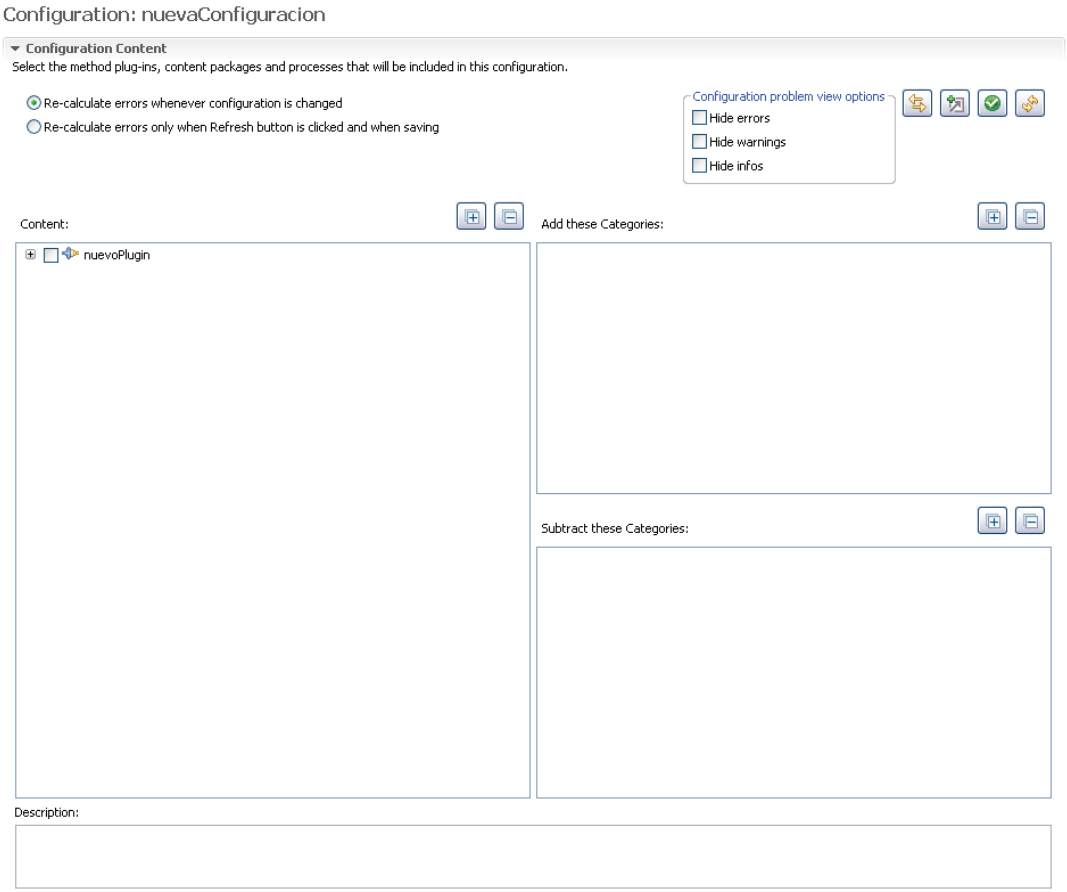

Figura 5.14 Selección de contenido para una *Configuration*.

Las *Configurations* permiten ser exportadas como sitio WEB independiente para generar versiones de lectura.

La definición de este sitio WEB generado, puede ser organizada en *Views* que serán representadas en un árbol de navegación.

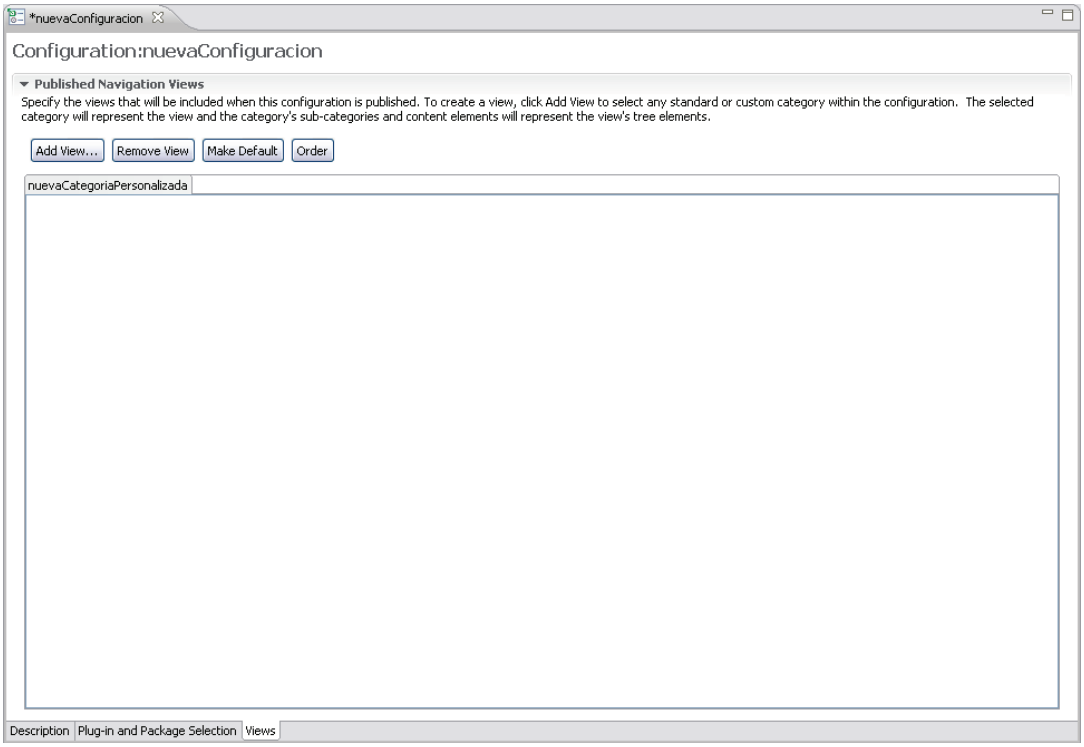

Figura 5.15 Ejemplo de *Views* de una *Configuration*.

Al generar la *Configuration* se solicitará que se especifique la carpeta en el disco duro dónde quedará el producto final.

Esta podrá ser publicada en un servidor WEB para su utilización.

5.3 Gestión de versiones

El modelado con *EPF composer* implica la edición de un proyecto subyacente con el modelo de la plataforma Eclipse.

Como es usual en la plataforma, está soportada la integración de gestores de versionado de código fuente como CVS, SVN, etc.

Para facilitar la edición concurrente del modelo fue utilizado el gestor SubVersioN integrable en el entorno mediante *plugins* disponibles en internet.

Luego de la prueba de dos plugins (Subclipse y Subversive), se optó por la opción de utilizar un cliente de SubVersioN independiente del entorno ya que las características del proyecto eclipse que gestiona el *EPF composer* podían generar errores y pérdida de información con el consiguiente re-trabajo, lo cual era precisamente lo que se intentaba evitar.

Los clientes SubVersioN utilizados fueron Tortoise SVN (para la plataforma Windows) y SVN Client (para la plataforma Linux).

6 Conclusiones y Trabajos Futuros

Debido a que este proyecto fue realizado en el marco de un trabajo de módulo de taller y para poder contar con el trabajo finalizado para el dictado del curso de Proyecto de Ingeniería de Software, fue que se debió dejar algunos puntos fuera del alcance.

A continuación detallamos dichos puntos:

Desagregación de las tareas en pasos:

Cómo puede verse desde el formulario de edición de tareas, una determinada tarea puede desagregarse en pasos de tareas. Esta información podría completarse para tereas complejas y que por lo tanto ameriten ser divididas en sub-tareas o pasos para llevarlas a cabo.

Creación de Extensiones:

La parte del MUM que fue especificada en SPEM 2.0 corresponde a "la parte básica de MUM", quedando por fuera las extensiones del modelo (Extensión Orientada a Objetos, GeneXus, PSP y Competisoft). Un agregado de valor sería realizar el pasaje de éstas extensiones a SPEM 2.0.

7 Referencias

[1] – SPEM 2.0: Software & Systems Process Engineering Meta-Model Specification. v2.0.
http://www.omg.org/spec/SPEM/2.0/
Último acceso: 06/08/2009

[2] – EPF: Eclipse Process Framework.
http://www.eclipse.org/epf/
Último acceso: 06/08/2009

[3] – MUM:
http://www.fing.edu.uy/inco/cursos/ingsoft/pis/memoria/experiencia2008/MUM/index.htm
Último acceso: 06/08/2009

[4] – Libro: An Integrated Approach to Software Engineering. 3º Edición.