FACULTAD DE INFORMATICA
UNLP

TESINA DE LICENCIATURA

TÍTULO: Plugin de Desarrollo basado en el metamodelo SPEM, utilizando el Proceso de Desarrollo de Software de COMPETISOFT.

AUTOR: Pérez, Solange Mariel.

DIRECTOR: Pesado, Patricia.

CODIRECTOR: Esponda, Silvia.

Resumen

El correcto desarrollo de un producto de Software debe seguir ciertas pautas de manera de cumplir con los objetivos planteados en las etapas iniciales del desarrollo. Además, debe ajustarse a los tiempos y costos estimados, para obtener los beneficios que se pretende alcanzar con su concreción.
A partir de esta aseveración, surge la necesidad de contar con un Proceso de Software correctamente definido.
Por esto, el presente trabajo se enfocó a poder brindar un Plugin de Desarrollo, donde se modela un Proceso que representa las actividades y productos involucrados en el Proceso de Desarrollo que define COMPETISOFT.
Para desarrollar el Plugin se estudió el metamodelo SPEM y para su implementación se utilizó la herramienta EPF Composer. Dicho desarrollo se complementó con el análisis de los nuevos conceptos de la Ingeniería de Procesos de Software.
La presente Tesis contiene una investigación enfocada principalmente en los tres puntos que se estudiaron para realizar el Plugin, ellos son el metamodelo SPEM, la herramienta EPFC, y el modelo COMPETISOFT.

Trabajos realizados

Se estudió el proyecto COMPETISOFT, para definir su Proceso de Desarrollo de Software utilizando la herramienta EPFC basada en SPEM 2, que permite documentar y describir los Procesos de Software.

Se investigó el uso de esta herramienta y su correspondencia con el metamodelo SPEM.

Se investigaron temas relacionados con Ingeniería de Proceso y el uso del metamodelado para entender la importancia de su aplicación en el desarrollo de software.

Se instanció un Plugin de Desarrollo, usando EPFC y basándose en el Proceso de Desarrollo de Software definido en COMPETISOFT. El Plugin permite contar con una aplicación estática, con posibilidad, incluso, de ser descargada de la Web, con el deseo de que las organizaciones puedan reutilizarla en sus proyectos.
Gracias a él, se puede también obtener diferentes vistas de la información según la configuración seleccionada para el trabajo.

Trabajos futuros

Durante el desarrollo de esta tesis se investigó el uso de la herramienta EPFC, se analizó el metamodelo de SPEM y todos sus posibles usos. También se analizó el Modelo de Procesos de COMPETISOFT centrándose en el Proceso de Desarrollo definido por el mismo.
Como futura investigación, se propone estudiar los restantes procesos definidos en COMPETISOFT a nivel de Categoría de Operación, como lo son el de Administración de un Proyecto Específico y Mantenimiento de Software, desarrollar un Plugin que los represente así como se hizo con el Proceso de Desarrollo, analizar también los diferentes niveles de capacidad de sus actividades y representarlo de forma de ser adaptado a las necesidades de cada organización.
Con este nuevo desarrollo se estaría obteniendo una representación completa del modelo de procesos de COMPETISOFT, así como también una publicación Web general de todo el proceso
Ampliar este Plugin para obtener una visión general y completa de todo el modelo de procesos de COMPETISOFT abarcando todas las áreas y definiciones posibles.

Conclusión

El modelado de un Proceso de Software es tan importante como su desarrollo. Modelarlo permite lograr la madurez del Proceso de Software, es decir poder definir, documentar, medir y mejorar continuamente.
El modelo de un Proceso de Software debe ser preciso, para poder gestionar el proceso de forma correcta, obteniendo un producto confiable y fácil de mantener, pudiendo realizar cambios en un futuro o durante el desarrollo, sin la necesidad de la redefinición total del Proceso.
Es por esto que se implementó este Plugin, que brinda la posibilidad de adaptación para cada proyecto particular según las necesidades y prioridades que se consideren más relevantes. Se utilizó y estudió el Proceso de Desarrollo implementado en COMPETISOFT, ya que se consideró interesante, por ser un modelo aplicable a las pequeñas y medianas empresa, y fácil de comprender y aplicar como un modelo de referencia. Además se eligió el metamodelo SPEM v.2.0 por la gran cantidad de ventajas que ofrece, al poder reutilizar componentes y brindar un repositorio donde almacenar los procesos, permitiendo su edición y recuperación de forma clara y fácil.
La obtención del Plugin, fue útil para la comprensión de los conceptos definidos en SPEM, ya que al instanciar el Proceso de Desarrollo de COMPETISOFT, se pudieron concretar muchas de sus definiciones y analizar profundamente otras.

Líneas de Investigación

Metamodelo SPEM

Herramienta EPF

Ingeniería de proceso

Estudio del Proyecto COMPETISOFT

Fecha de presentación: Noviembre 2009

Dedicada a las personas que estuvieron presente, siempre, en cada día de estos años de carrera.
A mamá, papá, Jesi y Sergio, que me brindaron su apoyo incondicional.

.

Índice

Índice de Ilustraciones

1. Introducción

El objetivo de toda organización que desarrolla productos de software, es obtener un producto con calidad que responda a los requerimientos que en un principio fueron planteados y a los tiempos que fueron estipulados para su desarrollo. Es muy importante para lograr este objetivo, tener en cuenta cuatro puntos claves: definir, medir, controlar y mejorar el proceso de la Gestión de Procesos de Software. [COM01]

Definir el proceso: La definición de un Proceso de Software crea el ambiente disciplinado y estructurado requerido para controlar y mejorar el proceso, además para definir cada proceso intrínsecamente se incluyen las responsabilidades de implementar y sostener el proceso en ejecución.

Medir el proceso: Las mediciones son la base para detectar las desviaciones del funcionamiento aceptable, además también son la base para identificar las oportunidades para mejorar el proceso.

Controlar el proceso: El control del proceso asegura que la variabilidad sea estable de modo que los resultados sean predecibles, esto significa mantener el proceso dentro de sus límites (inherentes) normales de funcionamiento. Involucra:

- Mediciones, es decir obtener información sobre el funcionamiento del proceso.
- Detección, analizar la información para identificar las variaciones en el proceso que se deben a causas identificadas.
- Corrección, tomar acciones para quitar la variación debido a las causas identificadas del proceso.

Mejorar el proceso: Aunque un proceso puede estar definido y bajo control, puede que no sea capaz de producir los productos que resuelven las necesidades del cliente ni los objetivos de la organización. Precisamente los procesos pueden ser mejorados realizando cambios que mejoren sus capacidades existentes o substituyendo subprocesos existentes por otros que sean más eficaces.

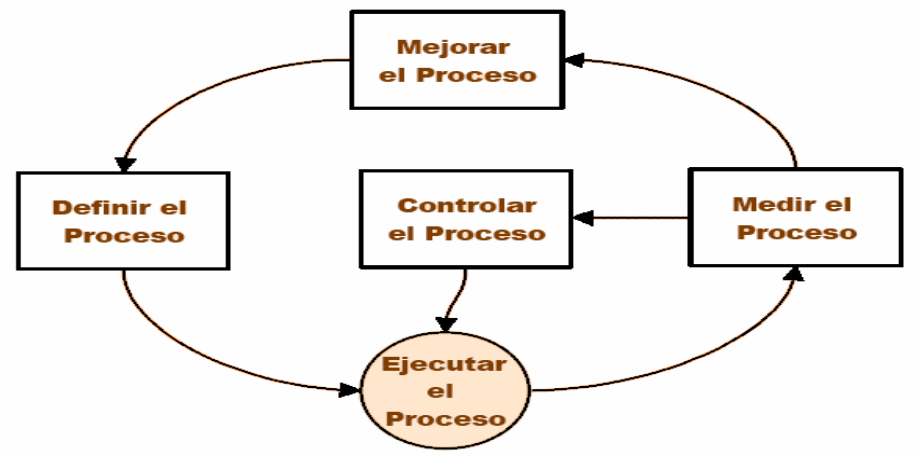

Ilustración 1. Niveles del Proceso de Ingeniería de Software. [COM01]

Los procesos controlados son procesos estables, y los procesos estables permiten predecir resultados, y por lo tanto lograr estimar posibles soluciones a problemas, que hacen a la eficiencia del desarrollo.
Es preciso definir qué es un proceso, y por eso se dice que siempre que para alcanzar algún fin deseado se necesite ejecutar una serie de acciones, y estas acciones tengan cierto orden, dependencias, roles responsables, resultados, tiempos de ejecución y herramientas de apoyo, se estará hablando de procesos. Estos procesos pueden ser predefinidos y personalizados adaptándose a las necesidades de cada proyecto. [WEB04]
Se define, entonces, como Proceso, al conjunto ordenado de pasos a seguir para llegar a la solución de un problema u obtención de un producto. En este caso particular, para lograr la obtención de un producto de software que resuelva un problema.

Estos pasos, si bien deben existir, son flexibles en su forma de aplicación, de acuerdo a la metodología o Proceso de Desarrollo escogido y utilizado por el equipo de desarrollo. El Proceso de Desarrollo de software se puede definir como el conjunto de actividades, métodos, prácticas y transformaciones que los individuos emplean para desarrollar y mantener el software. [REV]

1.1 Ingeniería de Procesos

Un elemento clave de la Ingeniería de Procesos es la aparición del paradigma que se basa en la Ingeniería Dirigida por Modelos.

Es muy importante encontrar en el ámbito de la modelización de procesos un lenguaje de modelado de procesos estándar que permita la descripción de los mismos, para así facilitar y unificar conceptos de forma tal que los desarrolladores no encuentren ambigüedad entre sus definiciones y de algún modo logren estandarizar sus métodos y metodologías de desarrollo.

Con este objetivo, han surgido metamodelos como SPEM, que tratan de proponer un formalismo de modelización de Procesos de Software. [ING01] Un Proceso de Software tiene una estructura jerárquica con varios niveles de agregación. Estos se definen como subprocesos (pueden ser opcionales), actividades y tareas (son la unidad atómica básica de trabajo). Los dos primeros junto con los procesos, tienen asociados un flujo de trabajo, todos estos componentes son los que se permiten modelar con metamodelos como SPEM.

Las actividades, por ejemplo, deben ser simples de aprender y usar, deben simplificar la comprensión del sistema, deben ser suficientemente poderosas para expresar la información requerida para modelar el sistema y deben ser lo suficientemente descriptivas para poder discutir el sistema sin ambigüedades.

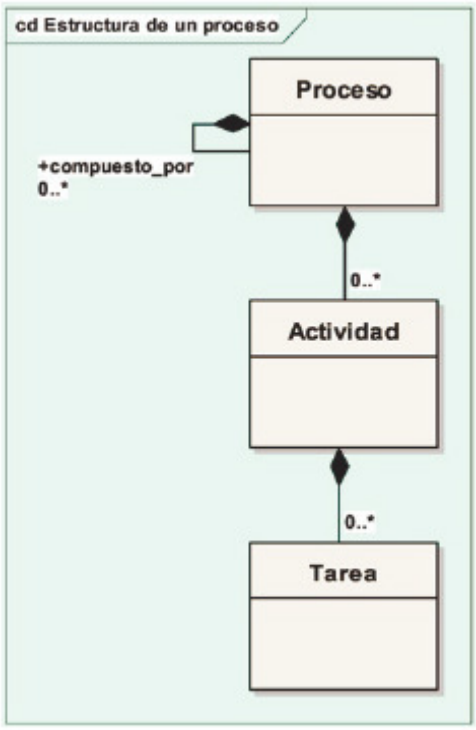

Ilustración 2. Estructura de un Proceso. [WEB02]

1.2 Modelos de Procesos de Software

Un modelo de Proceso de Software se describe como una abstracción o representación (textual, gráfica o formal) en la que se capturan los aspectos más importantes de un Proceso de Software. [UNI01]

Permite la representación de las actividades (describiendo sus tipos, propiedades y relaciones), recursos (todos los que se necesitan en los procesos), productos (describiendo sus tipos y propiedades) y actores (considerados como los roles de los desarrolladores, describiendo sus habilidades y responsabilidades) de un proceso, junto con las relaciones que se establecen entre estos componentes que definen, sus dependencias y ordenes de ejecución. Permiten mostrar la forma en que el proceso debería realizarse, brindando facilidad de comprensión y flexibilidad en la forma de trabajo. [GES01]

Estos modelos deben tener como características claves, la precisión, es decir, presentar una descripción exacta del sistema a modelar, deben poder ser comprensibles para quienes lo utilizaran, deben poseer un gran grado de abstracción, dando mayor importancia a los componentes más importantes y menor énfasis en los irrelevantes, además debe presentar un enfoque predictivo, para permitir deducir conclusiones sobre el sistema y por supuesto debe ser de menor costo y mayor sencillez en su construcción que el sistema que modela. Todo esto es requerido para poder encontrar errores o faltas del diseño antes de llegar a la etapa de implementaciones, donde el descubrimiento en esta etapa requerirá mayores costos y complicaciones, es decir, se encamina hacia el objetivo de minimizar riesgos.

Son entonces, los modelos, de vital importancia para la mejora de los procesos, logrando que estos sean supervisados, simulados, validados y verificados, permitiendo su estudio a menor costo sin emplear recursos o herramientas reales. [WEB03]

Sommerville define modelo de Proceso de Software como, una representación simplificada de un Proceso de Software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de Procesos del Software es una abstracción de un proceso real.

Los diferentes proyectos de software requieren de formas diferentes para llevar a cabo la resolución de sus problemas, por lo tanto, cada proyecto usará el modelo que considere más adecuado para sus necesidades. [SOM01]

El modelado es un medio para obtener los productos de software que son el objetivo final de un desarrollo, pero debe tenerse en cuenta que el modelado es un punto muy importante a considerar para lograr esta meta, considerando a éste, como un paso necesario para la comprensión y mejora continua de los procesos y para su evaluación. Esta última presenta como objetivo detectar los aspectos del proceso posibles a mejorar, pero para ello se necesita un marco efectivo, concreto y definido para la medición de los procesos y productos.

Tanto la evaluación como el modelado de los procesos deben integrarse para que las organizaciones alcancen un alto grado de madurez en sus procesos como lo indican los estándares ya definidos.

El modelado de procesos ha adquirido una importancia creciente como mecanismo que debe permitir, por un lado, una mejor comprensión de ese proceso con vistas a su evaluación y mejora y, por otro, la posibilidad de lograr un cierto grado de automatización del mismo.

Existen tres niveles de abstracción para modelar Procesos de Software:

- Nivel de proceso: consiste en el modelado de un proceso específico, por ejemplo el Proceso de Desarrollo de una aplicación.
- Nivel de modelos de procesos: consiste en el modelado de un proceso genérico (modelo de procesos) aplicable a una familia de procesos, por ejemplo el modelo de procesos RUP para el desarrollo.
- Nivel de metamodelos de procesos: consiste en el modelado de metamodelos de procesos, por ejemplo el metamodelo SPEM.

Los tres niveles se relacionan a través de los productos que generan, es decir, el resultado de un nivel, brinda la posibilidad de ejecución del siguiente, que utiliza como entrada el producto generado en el anterior. [VUL01]

SPEM es un metamodelo genérico para la definición de Procesos de Software y está basado en el metametamodelo universal de MOF (Meta Object Facility). Esta última, es una norma del OMG (Object Management Group) para la definición, representación y gestión de metadatos, al igual que UML, por lo que gana su inherente expresividad para representar modelos descriptivos de Procesos de Software.

1.3 MOF (Meta Object Facility)

El Desarrollo de Software Dirigido por Modelos es una rama de la Ingeniería del Software en la que los artefactos de software se representan como modelos para incrementar la

productividad, calidad y eficiencia económica en el Proceso de Software, donde un modelo proporciona una representación abstracta del código final de una aplicación. [MOM01]

En este campo, la iniciativa Model-Driven Architecture (MDA), patrocinada por la OMG, está constituida por una familia de estándares industriales, entre los que se destacan: Meta-Object Facility (MOF), Unified Modeling Language (UML), Object Constraint Language (OCL), XML Metadata Interchange (XMI), y Query/Views/Transformations (QVT).

Estos estándares proporcionan directrices comunes para herramientas basadas en modelos y para Procesos de Software dirigidos por modelos. Su objetivo consiste en mejorar la interoperabilidad entre marcos de trabajo ejecutables, en automatizar el Proceso de Desarrollo de Software y en proporcionar técnicas que ayuden a evitar posibles errores. [OMG01]

En este enfoque se requiere un metalenguaje de modelado que pueda ser utilizado en el ámbito MDA. Este lenguaje es MOF (Meta Object Facility).

MOF es un estándar de modelado, desarrollado por el OMG, por medio del cual se puede definir formalmente la sintaxis abstracta de su conjunto de constructores de modelos, además de proporcionar una semántica informal por medio del lenguaje natural. [VIC01]

1.3.1 Metamodelos

Un Metamodelo describe un conjunto de conceptos genéricos y sus interrelaciones, que sirven de base para la definición de Modelos de un cierto dominio. Por lo tanto, un metamodelo es un modelo de modelos.

Aplicando estas ideas al ámbito de los Procesos de Software, los modelos de Procesos de Software se construyen instanciando los conceptos del metamodelo de Procesos de Software genérico, es decir, por ejemplo de SPEM.

Esta instanciación es determinada por las características propias del modelo que se quiere elaborar. En el diseño de un modelo de Proceso de Software se deben respetar las relaciones entre los diferentes conceptos definidos en el metamodelo. [UNI03]

El metamodelado es la capa donde se define el lenguaje que sirve para especificar los modelos, dichos modelos están limitados al lenguaje en el que se generan. Los metamodelos son la definición de estos lenguajes.

El metamodelo, también es un modelo y debe estar escrito en un lenguaje bien definido. Define estructura, semántica y limitaciones de un conjunto de modelos.

La definición de metamodelo es un modelo con la capacidad de almacenar diferentes modelos. [INTI01]

MOF es un metamodelo de la OMG que permite especificar actividades en el Proceso de Desarrollo de Software.

1.3.2 Arquitecturas conceptuales MOF

MOF posee cuatro niveles conceptuales denominados M0, M1, M2 y M3.

M0 es el nivel de las instancias. Modela el sistema real, por ejemplo puede ser en un sistema de supermercados, un artículo llamado Aceite Patito 1 Lts.

M1 es el nivel de modelo de sistema concreto. Siguiendo con el ejemplo anterior sería definido el concepto de entidad como PRODUCTOS, atributos como NOMBRE, etc.

M2 es el modelo del modelo. Los conceptos de este nivel son las CLASES, ATRIBUTOS, VALORES, etc (metamodelo).

M3 el modelo que permite definir los elementos que constituyen los distintos lenguajes de modelado, por ejemplo el MOF, que fue definido por la OMG (meta-metamodelo). [WOR01]

En las ilustración 3 se encuentra la arquitectura conceptual MOF.

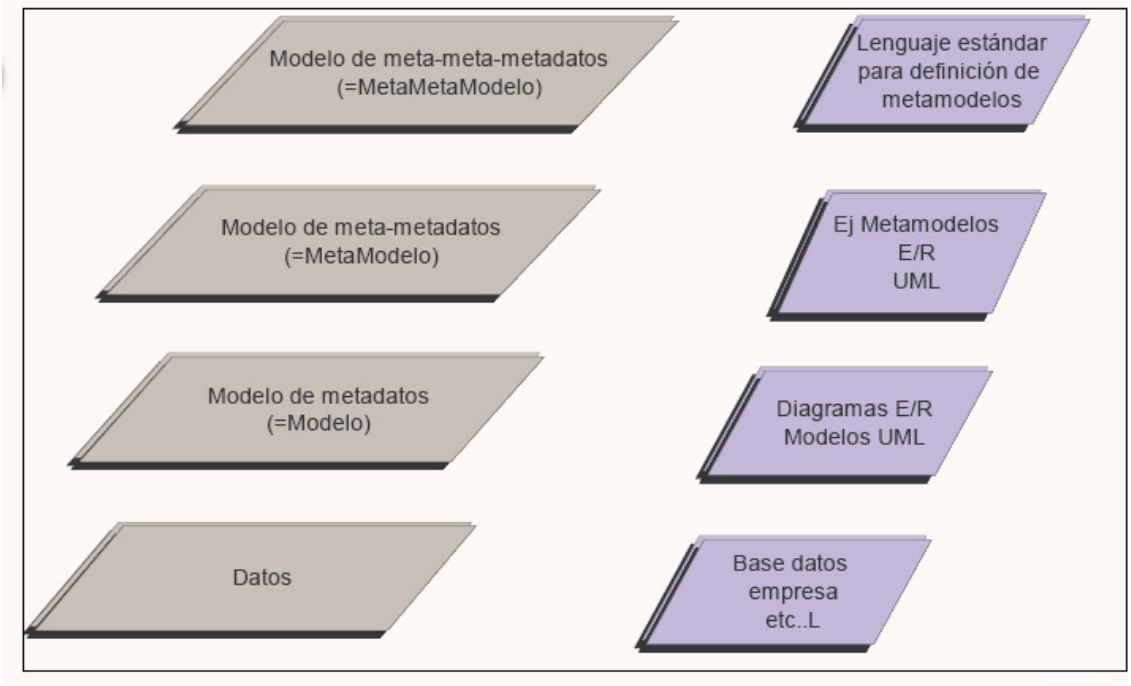

Ilustración 3. Arquitectura conceptual MOF. [UNI01]

SPEM2.0 está alineado con la tecnología estandarizada por el Object Management Group y por lo tanto sigue las guías del metamodelado estricto y la arquitectura de las cuatro metacapas. Cada capa se encuentra relacionada con la inmediatamente superior mediante la relación "instancia de" (instante of). En el nivel más alto de la arquitectura se encuentra MOF que es considerado el metamodelo de UML. La definición de UML se sitúa en la siguiente capa que es la capa donde se sitúan los metamodelos que se van a definir. En esta misma capa es donde se encuentra definido SPEM2.0. La siguiente capa realiza la instanciación de los elementos definidos en los metamodelos UML y SPEM, definiendo elementos como por ejemplo clase y asociación. La última capa es la capa de objetos donde se realizan las instancias de las clases e instancias de UML. [GES01]

1.3.3 Arquitectura y fases de desarrollo MDA

El Desarrollo dirigido por modelos pretende aumentar el nivel de abstracción y automatización para mejorar la productividad y calidad del software producido, y obtener una solución a los problemas, más rápida y eficiente.

El proceso MDA se basa en transformar modelos independientes de detalles de implementación en otros, que aportan los aspectos específicos de una plataforma concreta. Esta transformación se denomina transformación model-to-model y pasa de un modelo denominado PIM (Platform Independent Model) a otro denominado PSM (Platform-Specific Model) y una segunda transformación que traduce de modelo PSM a código. [MODEL01]

Se distinguen dos fases de este modelo:

Primera fase es la creación de un Modelo Independiente de la Plataforma (PIM), es el modelo de mayor nivel de abstracción del sistema, y describe la funcionalidad del sistema omitiendo los detalles de cómo y dónde será implementado.

Se representa el modelo de procesos de negocio a ser implementado sin hacer ninguna referencia a la forma que será desplegada la aplicación. A su vez, ignora los sistemas operativos, los lenguajes de programación, el hardware y la topología de red.

Comúnmente se usa UML o un derivado de UML para describir el PIM.

La segunda fase es la transformación de PIM, a uno o varios Modelos Específicos de Plataforma (PSM). Es el modelo PIM traducido a una plataforma específica. Un PIM puede generar múltiples PSMs, cada uno para una tecnología distinta. Generalmente, los PSMs deben colaborar entre sí para una solución completa y consistente. Los PSMs tienen que lidiar explícitamente con los sistemas operativos, los lenguajes de programación y las plataformas.

La fase que realmente aporta la independencia con la plataforma sobre la que se realizará el sistema es la fase II, ya que traduce del modelo genérico al modelo específico. [MDA01]

MDA es para los desarrolladores una nueva manera de organizarse y administrar arquitecturas empresariales basada en la utilización de herramientas de automatización de etapas, en el ciclo de desarrollo y servicios. Permite definir modelos y la transformación entre ellos, a partir de unos se generan otros de menor abstracción. [REU01]

1.3.4 Ventajas del uso de MDA

Las ventajas que presenta esta forma de trabajo son muchas, entre ellas MDA mejora la productividad, ya que hay una clara separación para los desarrolladores, que trabajan independientemente de los detalles de plataforma que se utilice, otra ventaja es la portabilidad que se centra en el desarrollo de las PIMs que son plataformas independientes, por lo tanto, todo lo que se especifique a nivel de PIM es portable. La interoperatividad es otra de las cosas que se mejoran, ya que múltiples PSM generados a partir de un mismo PIM pueden tener relaciones entre si, estas relaciones son llamadas MDA puentes. Cuando los PSM están orientados a distintas plataformas no se pueden comunicar directamente. Es necesario transformar conceptos de una plataforma en los conceptos de la otra (en eso consiste la interoperatividad). En MDA no sólo se generan PSM sino que también los puentes necesarios entre ellos.

El mantenimiento y documentación también son características que adquieren una amplia mejora, ya que centrarse en el PIM para luego realizar sus transformaciones trae como consecuencia que la documentación de alto nivel será consecuente con el código actual.

Puesto que está más tiempo “en el mercado” MDA ha sido estudiado, discutido y aplicado. Este aspecto facilita la aceptación de un método por parte de la comunidad de la Ingeniería del Software.

Como desventaja se puede mencionar que no proporciona suficientes guías metodológicas, sólo se define la estrategia general para la transformación PIM a PSM. [MDA01]

2. Motivación

Cuando se piensa en el desarrollo de un producto de software se debe tener muy presente la importancia que tiene el modelado del Proceso de Desarrollo que se seguirá, para lograr llegar al producto deseado bajo las características y condiciones que mejores resultados brinde.

Las consecuencias que se presentan una vez desarrollado un producto que no fue eficazmente modelado, medido y diseñado son muy graves e importantes, por ejemplo en cuestiones de tiempos mal estimados o costos mal estipulados.

Tradicionalmente la Ingeniería del Software se ha centrado en desarrollar métodos, técnicas, herramientas para aplicar los principios de Ingeniería al software. Pero en realidad, el foco ha estado centrado en el software como producto. Así, todas las técnicas, lenguajes, métodos, etc. están volcados en poder ser sistemático y riguroso al manejar los distintos artefactos que llevan al producto de software final. Un ejemplo claro es UML, un estándar para el modelado de los sistemas de software (productos).

En los últimos años, se ha desarrollado un área nueva, dentro de la Ingeniería del Software, que intenta hacer lo mismo pero enfocado en los procesos. Surge así SPE (Software Process Engineering) cuyo objetivo es "la definición, implementación, medición y mejora de los Procesos de Ingeniería del Software".

Un elemento clave ha sido la aplicación del nuevo paradigma de Ingeniería Dirigida por Modelos (MDE, Model Driven Engineering), de forma que el tipo de artefacto clave para trabajar con los Procesos de Software de manera sistemática y rigurosa son los Modelos de Procesos (MP). Ahora, los modelos pasan a jugar un papel más importante.

En esta línea, recientemente OMG, el consorcio industrial promotor de UML, ha publicado su nuevo estándar SPEM (Software & Systems Process Engineering Metamodel Specification). Dicho estándar establece los elementos clave para la representación de métodos, ciclos de vida, técnicas, roles, actividades, procesos, metodologías, plantillas, etc. en Ingeniería del software.

Disponer de los modelos de Procesos de Software tiene una ventaja especial de cara a la certificación: es un requisito para poder alcanzar los niveles 2 (proceso definido) y 3 (proceso gestionado) de CMMI. Igual ocurre con las certificaciones de ISO. Pero además, se abren nuevas perspectivas que se derivan de que los procesos se manejan (representan, definen, almacenan, publican, etc.) de una manera mucho más eficiente.

Algunas posibilidades de la tecnología SPEM son:

- Poder integrar varios PS, cada uno con su modelo.
- Dar soporte a la mejora de PS.
- Facilitar la evolución del modelo de un PS.
- Gestionar de forma integrada el proceso y su ciclo de vida (diseño, despliegue, ejecución, automatización, mejora, etc.).
- Facilitar la construcción de plataformas más potentes de cara a la gestión de procesos y la gestión de proyectos.

- Permitir que el repositorio de procesos sea más genérico y tenga más capacidad semántica. Esto significa facilitar técnicamente la gestión del conocimiento relacionado con los procesos.

Por el conocimiento del Proyecto COMPETISOFT inserto en el proyecto de investigación del Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo (CYTED) y en el que participaron universidades, empresas, centros públicos y organismos de estandarización de trece países, se eligió, para realizar el Plugin, el Proceso de Desarrollo de Software de COMPETISOFT Perfil Básico.

Dicho proceso permite ayudar al usuario a implementar el desarrollo de software, respetando los tiempos y costos estipulados en un comienzo, y así lograr una mejora en el mismo.

3. Investigación previa

3.1 SPEM 2.0 Software and System Process Engineering Metamodel Specification

El estándar SPEM (Software & Systems Process Engineering Metamodel) versión 2.0, fue desarrollado y aprobado por la OMG (la organización industrial promotora de UML) y tiene como objetivo ser el estándar industrial para la representación de modelos de Procesos de Ingeniería del Software e Ingeniería de Sistemas.

Su alcance abarca los elementos mínimos necesarios para definir dichos procesos, sin añadir características específicas de un dominio o disciplina particular; pero sirve para métodos y procesos de diferentes estilos, culturas, niveles de formalismo, o modelos de ciclos de vida. [GUIA]

SPEM no es un lenguaje de modelado de procesos en general, ya que está orientado a los Procesos de Software, y permite brindar mayor facilidad de comprensión y comunicación humana en las organizaciones. Posibilita, además, la reutilización en los procesos, contribuyendo gracias a todo esto a la mejora de procesos y a su automatización.

El metamodelo SPEM permite abstraerse de las características particulares de cada Proceso de Software.

SPEM ofrece un framework para el modelado, dando una sintaxis y una estructura común para cada aspecto del proceso, incluyendo: roles, tareas, artefactos, lista de chequeo, etc.

SPEM presenta características que son importantes destacar:

Permite hacer una separación clara entre la definición de métodos y su aplicación al desarrollo de procesos concretos. En primer lugar se define un “Method Content” (contenido de método) con sus “Content Element” (elementos de método) y luego se combinan y reutilizan los elementos ya definidos, para obtener los “Process” (procesos).

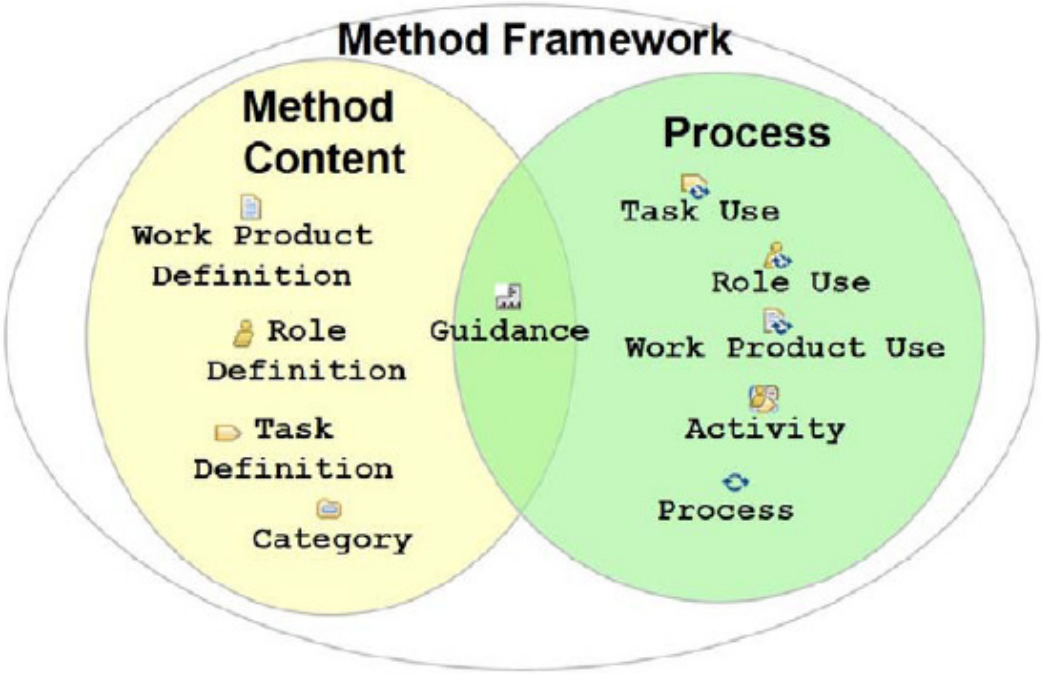

Ilustración 4. Elementos principales del estándar SPEM. [WEB02]

La reutilización y organización se logran con la organización en paquetes que ofrece el metamodelo. Cada paquete es una unidad lógica que extiende otros paquetes.

En SPEM se puede observar la clara separación entre el paquete "Method Content", donde se especifican los métodos y en el cual se definen las tareas, con sus pasos, sus entradas y salidas y los roles correspondientes, y el paquete, "Process", cuyo principal elemento es la actividad, que se forma por otras actividades, tareas, etc. que se definen dentro de un espacio de nombres y para los que se han descrito una serie de relaciones específicas para un método o proyecto.

Se ve también que en la intersección de estos dos grandes conjuntos se encuentran las guías (plantillas, ejemplos) que asisten dando soporte a ambos conjuntos.

SPEM ofrece también mecanismos para que los procesos sean variables y extensibles. Se definen "Methods Plug-in" mediante los cuales se pueden personalizar los métodos sin modificar su contenido original.

El paquete "Method Plug-In" proporciona las capacidades para gestionar librerías de "Method Content" y "Process" permitiendo mecanismos de variabilidad y extensibilidad mediante la reutilización de "Methos Content" y "Process" determinados.

Al crear un "Method Plugin", SPEM 2.0 permite referenciar a otros Plug-ins cuyo contenido se quiere reutilizar. [GUIA]

Un Plugin, en este caso, hace referencia a un módulo de software que añade nuevas características a un sistema más grande. Este módulo permite conectarse al sistema principal, sin inconvenientes, anexándose a éste, para aumentar su funcionalidad, sin afectar la función original. Del mismo modo, SPEM 2.0 habilita los mecanismos de contribución, reemplazo o supresión entre Plug-ins.

Otra característica distintiva de SPEM, es que brinda la posibilidad de modelar procesos que pertenezcan a diferentes modelos de ciclos de vida, como son el modelo en cascada, modelo de iteraciones, incremental, etc. Además permite definir Patrones de procesos reutilizables, que son bloques utilizados para crear nuevos Procesos de Software y mejores prácticas para un ensamblado rápido de los procesos.

Existen dos tipos principales de patrones de procesos, los "Capability Patterns" y los "Delivery Process".

Los primeros son procesos especiales que describen conjuntos de actividades comunes en determinadas áreas de los procesos. Los "Capability Pattern" pueden aglutinar mejores prácticas de desarrollo de procesos para disciplinas, tecnologías o estilos de desarrollo.

Los "Delivery Process" por otro lado, son procesos que cubren el ciclo de vida completo del proceso desde el comienzo hasta el final. Un "Delivery Process" proporciona el modelo de ciclo de vida completo con fases predefinidas, iteraciones y actividades.

SPEM es un marco de trabajo conceptual además de un metamodelo para Ingeniería de Procesos. Este marco de trabajo consiste en normalizar la representación y gestionar un repositorio de contenido de métodos reutilizables. Para luego configurar un marco de trabajo con procesos integrados y adaptados para cada proyecto particular.

Hasta aquí se ha dado una breve descripción de los componentes y características que ofrece SPEM para el metamodelado, a continuación se detallan y amplían todos los conceptos relacionados a este metamodelo.

SPEM2.0 es la tercera especificación definida para el metamodelo de Ingeniería de Procesos de Software y Sistemas. Anteriormente se implementaron tanto la especificación SPEM1.0 como SPEM1.1 que incorporaba pequeñas actualizaciones respecto a la anterior.

SPEM2.0 es una versión mejorada, solucionando problemas básicos de versiones anteriores. Además esta nueva especificación es compatible con la especificación MOF2.0 (Meta Object Facility).

Desde la creación del estándar SPEM1.1, pocas han sido las implementaciones que se han llevado a cabo basadas en esta especificación del metamodelo. Una de las razones principales de la falta de éxito entre los implementadores, ha sido la ambigüedad semántica del metamodelo y la dificultad añadida que implicaba para los usuarios del mismo.

SPEM1.1, se define por una parte como un metamodelo por sí mismo creado a partir de UML1.4, así como un perfil de UML.

El nuevo estándar UML2.0, incluye características como técnicas de modelado mejoradas y capacidad de intercambio de gráficos, aportes ambos que indiscutiblemente suponen importantes ventajas para el modelado de procesos. Es por ello que uno de los objetivos principales solicitados por el OMG a la hora de presentar propuestas fue que el nuevo estándar SPEM2.0 fuese compatible con UML2.0, que ha sido diseñado con un carácter más modular que su antecesor UML1.4.

Resumiendo, se puede destacar que los problemas principales que se encontraron en el estándar SPEM1.1 fueron, que se basaba en UML1.4, en lugar de UML2.0 y que tenía ambigüedad semántica y era difícil su comprensión. [VUL01]

3.1.1 SPEM 2.0

Los requisitos de carácter general que cumple el nuevo Metamodelo SPEM2.0, así como los objetivos de mejora respecto a SPEM1.1 son los siguientes:

- Compatibilidad con UML2.0.
- Guías para usuarios no expertos, y aumento de la usabilidad y funcionalidad de la especificación.
- Guías para migración del modelo de procesos de SPEM1.1 a SPEM2.0.
- Desarrollo de extensiones para conseguir la automatización de procesos.
- Clara separación entre la definición de los métodos y la aplicación de dichos métodos al desarrollo de un proceso concreto.
- Mantenimiento consistente de distintas alternativas de procesos.
- Soporte para distintos modelos de ciclo de vida.
- Mecanismo flexible para dar soporte a la variabilidad y extensibilidad de los procesos.

- Ensamblado rápido de procesos mediante el uso de patrones.
- Utilización de los principios de la encapsulación para conseguir componentes de proceso reemplazables y reusables.

La nueva definición del metamodelo SPEM debe cumplir con ciertos objetivos propuestos, entre ellos la posibilidad de poder modelar cualquier Proceso de Software. Además el metamodelo obtenido debe ser un metamodelo de MOF2.0, incluyendo un esquema XML más rico y controlado que el de la versión anterior, debe ofrecer también compatibilidad con los modelos ya definidos con anterioridad en las versiones viejas de SPEM. [VUL01]

3.1.2 Mecanismos de trabajo y posibles escenarios

SPEM está definido como metamodelo y como perfil UML2.0, presentando así dos mecanismos de trabajo posibles:

Metamodelo MOF-Compliant

SPEM 2 está basado en MOF (Meta Object Facility), que define una arquitectura de modelado de cuatro niveles conceptuales.Este mecanismo de trabajo se refiere a un metamodelo MOF del nivel M2, que define sus elementos mediante instanciación de elementos del meta-metamodelo universal, del nivel M3. SPEM está dentro del nivel M2, por ejemplo, conceptos de SPEM como “Actividad”, “Producto de Trabajo”, o las relaciones “Actividad precede Actividad”, “Producto de Trabajo es entrada de Actividad”, “Métrica tiene una Unidad de Medida” o “Tabla está compuesta por Atributos” son instancias de asociación-MOF.Usando este mecanismo descripto anteriormente, SPEM define instancias concretas de estos conceptos, como son, el rol Analista en Sistemas, el artefacto Caso de Uso, etc.

Perfil UML2

SPEM se entiende como un conjunto de estereotipos UML2 que permiten representar métodos y procesos utilizando UML 2 (que incluye características como técnicas de modelado mejoradas y nuevas capacidades de intercambio de gráficos). En este caso, la definición sólo abarca la presentación, mientras que las definiciones semánticas y restricciones están en el metamodelo anterior.

Los estereotipos incluidos en el perfil UML 2 de SPEM incluyen íconos para la representación visual. Esto permite utilizar diagramas UML con dichos estereotipos para representar los métodos y procesos.

Como ventaja de este perfil, se destaca que no necesita desarrollar herramientas CASE especiales para crear y mantener los métodos y procesos, ya que se puede utilizar cualquier herramienta genérica de modelado UML.

Con ambos mecanismos se puede crear biblioteca de métodos, que contengan elementos de métodos y fragmentos de procesos. [UNI01]

SPEM es un metamodelo para Ingeniería de Procesos, pero además es un marco de trabajo conceptual, que ofrece conceptos usados para modelar, documentar, presentar, publicar, gestionar, intercambiar y realizar métodos y Procesos de Software. Existen dentro de este marco de trabajo diferentes escenarios de uso, entre los más habituales se encuentran:

1) Se puede trabajar con un conjunto organizado de roles, tareas, guías, productos de trabajo, fragmentos de métodos y procesos, los cuales forman un repositorio de "Contenidos de Métodos" que pueden ser reutilizables. Todos estos conceptos sobre procesos, son agrupados ya que son necesarios para que los desarrolladores los tengan en cuenta a la hora de comprender que tareas deben definirse, con que roles, con que entradas y salidas, cuales son las funciones de cada rol, como se clasifican las tareas según su objetivo, como obtener las guías o referencias para trabajar ordenada y adecuadamente.

2) También, es posible, combinar y reutilizar los elementos de métodos definidos anteriormente para configurar procesos que guiarán los proyectos, darán soporte al desarrollo, gestión y crecimiento de Procesos de Software, es decir de la creación de procesos elementales, llamados patrones de procesos, se puede originar un proceso completo o una metodología, que los incluya a todos.

3) SPEM permite dar soporte al despliegue del contenido de método y proceso que se requieran en cada caso particular que se presente, teniendo en cuenta la diversidad de proyectos en la organización y la escasa probabilidad de que un proyecto se ejecute dos veces. En este punto es importante recordar que el nivel 3 de CMMI (Capability Maturity Model Integration), correspondiente a un proceso definido, necesita disponer de procesos estándares en la organización y de mecanismos de particularización y SPEM 2 provee de ambas cosas.

Entre otras capacidades adicionales, SPEM 2 incorpora conceptos para:

- Reutilización de procesos o patrones de procesos.

- Variabilidad (procesos que incluyen partes alternativas configurables).

- Particularización (los usuarios definen sus propias extensiones, omisiones y puntos de variabilidad sobre procesos estándares reutilizados).

4) Generar plantillas para planes de proyecto concretos, contando así con la posibilidad de poder disponer de mayor cantidad de información y mejor disponibilidad de ésta, de manera automática, a la hora de definir los planes de los proyectos. Para darles auténtico valor, las definiciones de los procesos deben ser desplegadas en formatos que permitan su realización automática. Para ello, SPEM 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 (repetición, una o varias veces en un proyecto, de una o varias definiciones de trabajo) y las ocurrencias múltiples (varias instancias de una definición de trabajo pueden llevarse a cabo a la vez de forma paralela). [UNI01]

3.1.3 Estructura del Metamodelo

La estructura del metamodelo SPEM2.0 está compuesta por 7 paquetes de metamodelado. Cada uno de estos paquetes es una unidad lógica que extiende los paquetes de los que depende, agregando estructura o funcionalidad.

Esta estructura se puede adaptar a las necesidades de modelado que se consideren convenientes en cada situación particular, haciendo una implementación parcial de SPEM 2, pudiendo trabajar con paquetes de niveles inferiores sin tener en cuenta los paquetes superiores.

Se puede elegir utilizar diferentes niveles de capacidades, conjuntos de conceptos, y niveles de formalismo para expresar sus procesos utilizando unos u otros paquetes.

Los paquetes que componen la estructura son los siguientes:

1) Core

2) ProcessStructure

3) ManagedContent

4) ProcessBehaviour

5) MethodContent

6) ProcessWithMethods

7) MethodPlugin

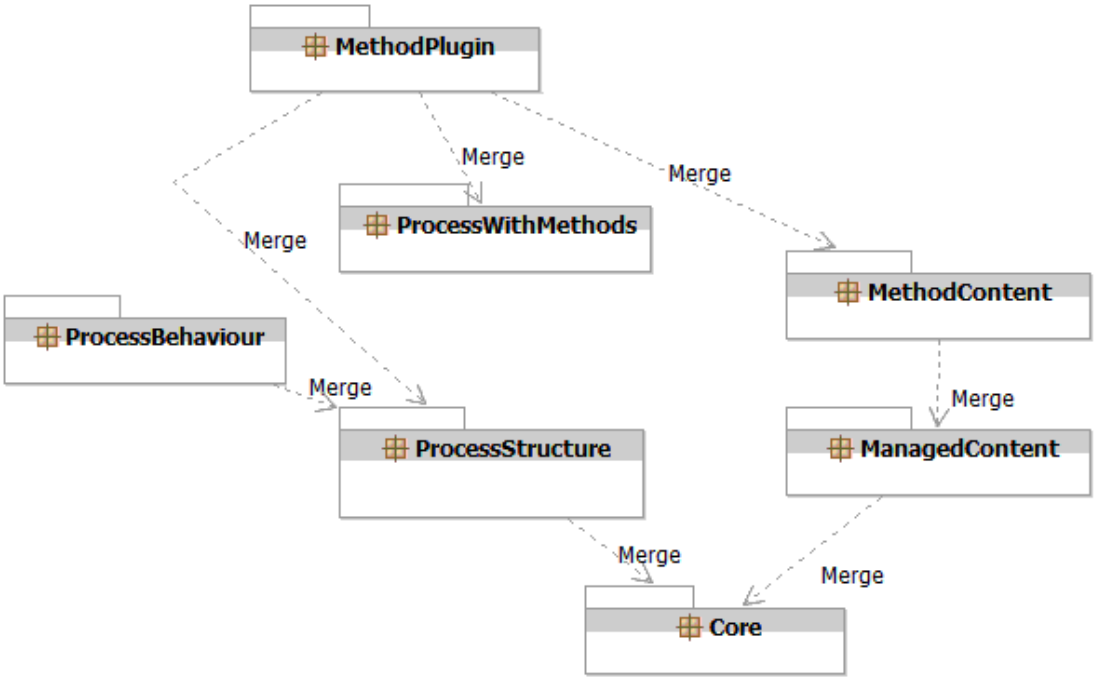

Ilustración 5. Estructura de SPEM. [VUL01]

En muchos casos, las clases del metamodelo, son definidas de forma simple en unidades de niveles inferiores y luego son extendidas a unidades superiores mediante, el mecanismo de combinación de paquetes, con propiedades y relaciones adicionales para poder cumplir requisitos más complejos de modelado de procesos.

A continuación se detallan las capacidades que brinda cada paquete:

Core: El paquete "Core" o núcleo del Metamodelo SPEM 2.0 contiene todas las clases y abstracciones que constituyen la base para el resto de los paquetes del metamodelo. Todas las clases comunes a todo el metamodelo están incluidas aquí.

Las superclases principales son "Extensible Element", "Kind", "Parameter Direction Kind", "Work Definition", "Work Definition Parameter" y "Work Definition Performer Map".

Las 3 últimas son las más importantes y se describen brevemente a continuación:

SUPERCLASE	DESCRIPCION
Work Definition	Clase abstracta que permite generalizar cualquier tipo de trabajo dentro de una especificación en SPEM 2.0. Es uno de los elementos más importantes de la especificación.
Work Definition Parameter	Generalización de los elementos del proceso que representan parámetros de entrada o de salida para un Work Definition determinado.
Work Definition Performer Map	Clase abstracta que permite generalizar la relación entre aquel que realiza un trabajo y un Work Definition

Ilustración 6. Superclases del paquete CORE.

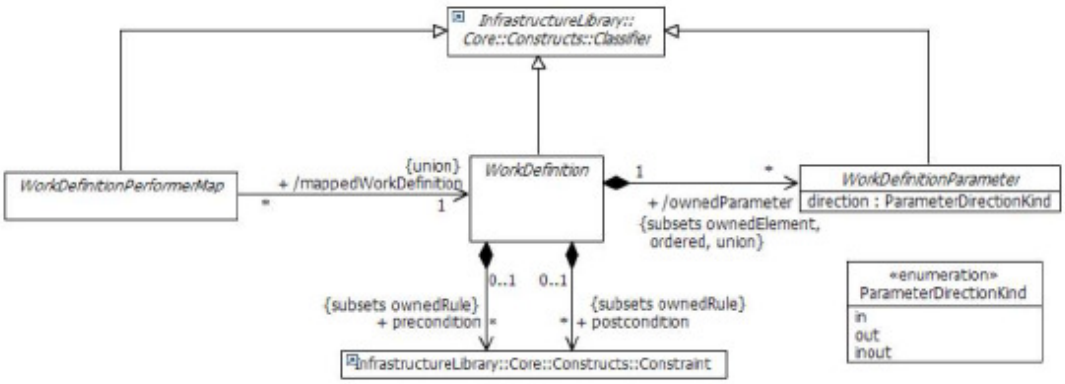

Ilustración 7. Paquete Core. [LEN01]

Process Structure: Contiene las clases necesarias para la creación de modelos de procesos. Soporta la creación de modelos de procesos simples y flexibles, es decir, define la estructura de desglose de trabajo estática mediante anidamiento de actividades y dependencias de precedencia entre ellas. El elemento principal es la Actividad, que a su vez puede descomponerse en otras actividades y contener otros tipos de elementos como la lista de Roles que realizan cada actividad y los Productos de Trabajo que son entradas y/o salidas. También provee capacidades para reutilización mediante ensamblado de procesos usando conjuntos de actividades enlazadas de forma dinámica.

Las clases más relevantes de este paquete son:

CLASE	DESCRIPCION
Activity	Elemento que sirve para representar la unidad básica de trabajo o para representar un proceso en sí mismo.
Activity Use Kind	Elemento usado para reusar una actividad relacionándola con otra.
Breakdown Element	Clase abstracta que sirve para generalizar cualquier tipo de elemento que forma parte de una actividad de nivel superior.
MileStone	Evento significativo dentro del Proceso de Desarrollo.
Process Element	Clase abstracta que generaliza cualquier elemento que forma parte de un proceso expresado en SPEM 2.0.
Process Parameter	Especialización de Work Definition Parameter y Breakdown Element que sirve para representar entradas y salidas de un proceso.
Process Performer Map	Elemento que representa las relaciones entre una actividad y las instancias de Role Use que indican que el Role Use participa de alguna manera en el trabajo desempeñado en la Activity.
Process Responsibility Assignment Map	Elemento que representa una relación entre instancias de Role Use y un único Work Product Use.
Role Use	Elemento que sirve para representar quién realiza o participa en una Activity.
Work BreakDown Element	Especialización de Breakdown Element que se usa para representar alguna clase de trabajo.
Work Product Use	Especialización de Breakdown Element que se usa para representar o bien cualquier entrada o salida de una actividad o bien cualquier participante en la actividad.
Work Product Use Relationship	Elemento que sirve para expresar relaciones (que pueden ser de distintos tipos) entre Work Products.

Work Sequence	Relación entre dos Work Breakdown elements que sirve para expresar que el comienzo de uno depende del final del otro.
Work Sequence Kind	Elemento que sirve para expresar relaciones de orden entre dos Work BreakDown Element de manera más general que en el elemento anterior.

Ilustración 8. Clases del paquete Process Structure.

En la siguiente figura, se muestran las clases y sus relaciones en la estructura Process Structure:

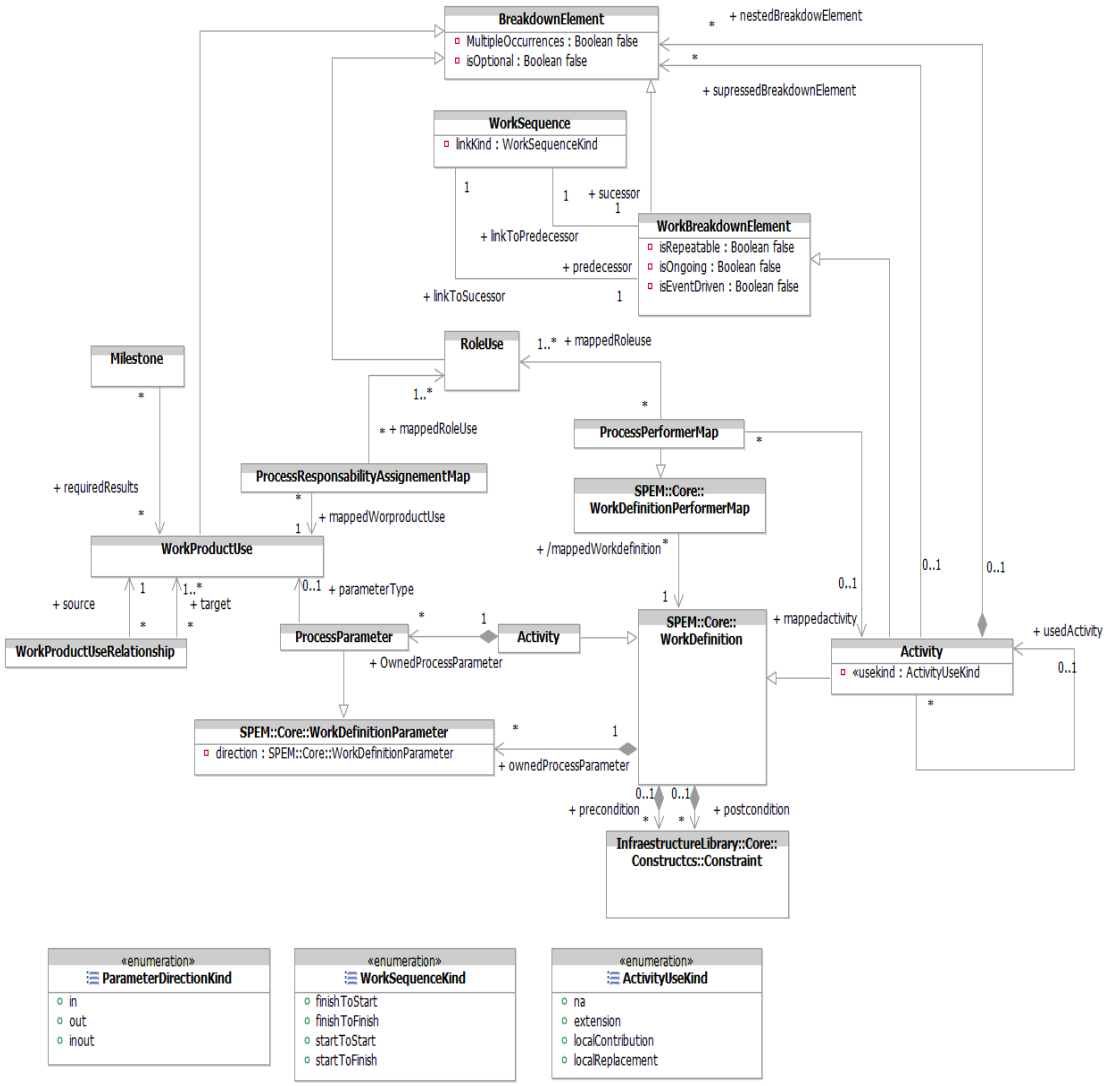

Ilustración 9. Estructura ProcessStructure. [VUL01]

Process Behaviour: Permite representar la parte dinámica de los procesos, su comportamiento. Permite extender las estructuras del paquete anterior con modelos de comportamiento externos: diagramas de actividad de UML 2 (comportamiento de proceso), máquina de estados (ciclo de vida de un producto de trabajo), etc. En vez de incluir un mecanismo propio para representar el comportamiento, se opta por reutilizar los ya existentes (de UML o de otro tipo).

Managed Content: Este paquete permite dotar a los procesos o sistemas de anotaciones y descripciones que pueden ser expresadas como modelos y que por lo tanto deben ser documentadas y gestionadas como descripciones del lenguaje natural. Esta posibilidad es muy interesante y útil porque ciertos conceptos no pueden ser formalizados con modelos, sólo pueden ser capturados con documentación en lenguaje natural. Hay libertad total para combinar modelos estructurales de proceso con contenidos en lenguaje natural. Así, un proceso puede estar formado sólo por una colección de guías definiendo buenas prácticas (esto es especialmente apto en el caso de métodos ágiles poco estructurados); sólo una estructura de actividades sin ningún tipo de documento textual; o una combinación interrelacionada de ambas cosas.

El elemento principal de este paquete es la clase abstracta "Describable Element" que es la superclase de todos los elementos del sistema para los cuales es posible tener una descripción textual. Además de esa clase, que va a permitir describir (de una u otra manera) todos los elementos del modelo de proceso, los otros componentes fundamentales del paquete son:

COMPONENTES	DESCRIPCION
Category	Componente que sirve para clasificar en categorías los distintos elementos.
Content Description	Componente usado para almacenar descripciones textuales de un Describable Element.
Guidance	Elemento que proporciona información adicional sobre cualquier Describable Element. Pueden ser de distintos tipos guías, plantillas, checklists etc.
Metric	Elemento que define medidas sobre Describable Elements.
Section	Componente utilizado para estructurar en distintas partes las descripciones de los elementos.

Ilustración 10. Componentes del paquete Managed Content.

Method Content: Contiene los conceptos de SPEM 2.0 relacionados con los usuarios y la organización. Estos conceptos son necesarios para construir una base de conocimiento sobre desarrollo que pueda ser utilizada independientemente del proceso o proyecto específico. Los principales elementos de método se derivan del patrón básico de SPEM: alguien (Rol) hace algo (Tarea) para obtener algo (Producto de Trabajo) basándose o ayudándose en algo (Guía). Por tanto, los elementos de método permiten describir, con el detalle necesario, cómo se alcanzan los objetivos del proceso haciendo qué tareas, por qué roles, usando qué recursos y obteniendo qué resultados. Los procesos reutilizan y relacionan entre sí los elementos de método de distintas maneras para diferentes tipos de proyectos. Su objetivo principal es definir las tareas, organizarlas en distintos pasos, definir cuales son los productos de entrada y salida de cada una de ellas y especificar quien ha sido el que ha realizado la tarea.

Los principales componentes de este paquete son:

COMPONENTE	**DESCRIPCION**
Default Responsibility Assignment Map	Componente utilizado para especificar relaciones entre instancias de Role Definition y Work Product Definition.
Default Task Definition Parameter	Especialización de Work Definition Parameter que permite usar Work Product Definitions como un atributo opcional.
Default Task Definition Performer Map	Elemento usado para relacionar instancias de Role Definition con instancias de Task Definition.
Method Content Element	Generalización de cualquiera de los elementos que van a componer un método.
Optionality Kind	Enumeración para describir si los Task Definition Parameters son opcionales u obligatorios.
Qualification	Elemento utilizado para documentar las habilidades o competencias que deben poseer los Role Definitios.

Role Definition	Conjunto de habilidades, competencias y responsabilidades de un participante o de un conjunto de participantes.
Step	Elemento usado para organizar una Task Definition en partes o subunidades de trabajo.
Task Definition	Componente que define el trabajo llevado a cabo por las instancias de Role Definition, es decir una unidad asignable de trabajo.
Tool Definition	Especificación de la participación de una aplicación (de cualquier tipo) para la realización de una tarea.
Work Product Definition	Cualquier cosa que es consumida, producida o modificada por las tareas. Constituye la unidad básica para conseguir la reutilización.
Work Product Definition Relationship	Componente que prácticamente tiene la misma semántica que Work Product Use Relationship sin embargo se utiliza para representar relaciones por defecto entre Work Products que han sido definidas en las descripciones generales de los métodos.

Ilustración 11. Componentes del paquete Method Content.

El objetivo principal de este paquete es el de definir tareas ("Task Definition"), organizarlas en distintos pasos ("Steps"), definir cuáles son los productos de entrada-salida de cada una de ellas ("Work Product Definition") y especificar quién ha sido el que ha realizado dicha tarea ("Role Definition").

En la siguiente figura se muestra Method Content:

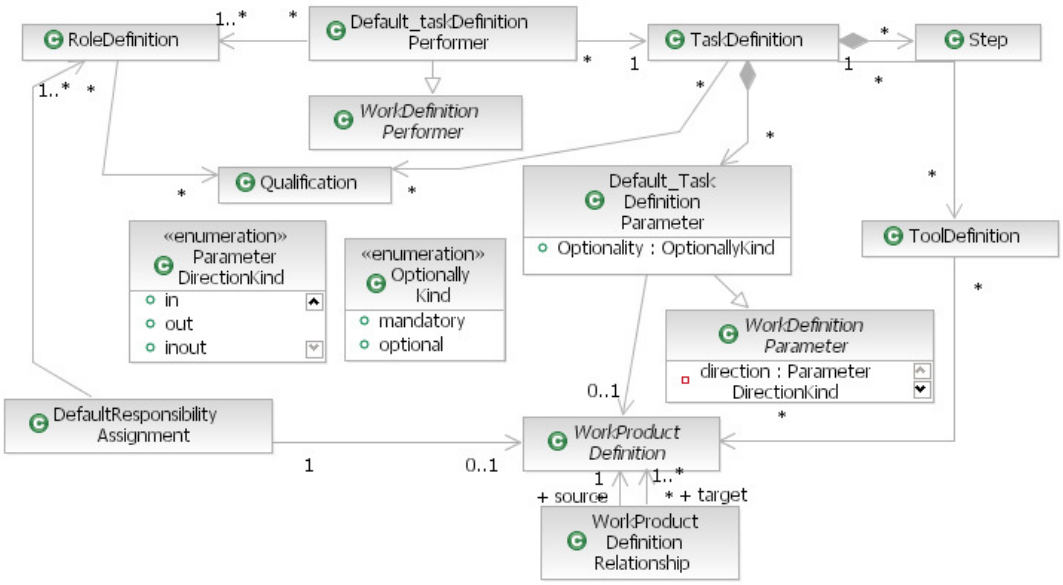

Ilustración 12. Paquete Method Content. [VUL01]

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

Los "Describable Element" son elementos de método (o de proceso) que pueden tener descripciones textuales mediante una Descripción de Contenido ("Content Derscription") opcional, que incluye nombre de presentación, descripción breve, descripción principal (permite texto enriquecido de diferentes formatos, imágenes, etc.), y propósito. Un "Describable Element" puede tener cero o muchas Secciones ("Section"), que a su vez pueden estar anidadas.

Además, puede estar asociado con, cero o muchas, Guías ("Guidance") y con, 0 o muchas, Categorías ("Category"). Estas últimas sirven para poder clasificarlos y agruparlos de múltiples formas.

Process With Methods: Contiene los elementos necesarios para integrar el paquete "Process Structure" con los conceptos y elementos del paquete "Content Method". Al asociar elementos de método a partes específicas de procesos, se crean nuevas clases (tarea en uso, rol en uso, etc.) que heredan de los elementos de método pero pueden tener cambios individualizados.

El paquete "Process With Methods" proporciona las estructuras de datos necesarias para reflejar las buenas prácticas de la industria y permitir la reutilización entre distintas instancias de métodos y procesos.

Sus componentes se describen a continuación:

COMPONENTE	**DESCRIPCION**
Activity	Grupo de distintos elementos (otras actividades, task uses, roles, milestone etc) definidos en un espacio de nombres y

	para los que se han descrito una serie de relaciones según el método o proyecto en el que se encuentre.
Breakdown Element	Generalización de Process Element que es parte de una estructura compuesta por varios elementos.
Composite Rol	Role Use especial que referencia a mas de un Role Definition con el objetivo de simplificar.
Method Content Kind	Refinamiento de Kind (del paquete Core) para Method Content Element.
Method Content Package	Elemento que describe un paquete que se caracteriza por contener únicamente Method Content Elements.
Method Content Packageable Element	Cualquier elemento que puede ser empaquetado en un Method Content Package.
Method Content Use	Generalización para Breakdown Elements que hace referencia a un Method Content Element concreto. Es el concepto clave para entender la separación entre Process y Method Content. Los Method Content poseen una serie de elementos y una serie de relaciones que pueden ser modificados para un Process particular para el cual ha sido creado el Method Content.
Planning Data	Process Element que añade información sobre Planificación.
Process Kind	Refinamiento de Kind (paquete Core) para referirse a los Process Element.

Process Package	Un paquete que únicamente puede contener Process Elements.
Process Packageable Element	Cualquier elemento que puede ser empaquetado dentro de un Process Package.
Process Performer Map	Elemento que extiende el Process Performer Map del paquete Process Structure, añadiéndole una asociación adicional con Task Use.
Role Use	Un role en el contexto de una actividad determinada.
Task Use	Task Definition dentro del contexto de una actividad determinada.
Team Profile	Agrupación con estructura jerárquica de Role Uses o Composite Roles.
Work Product Use	Work Product Definition dentro del contexto de una actividad determinada.

Ilustración 13. Componentes del paquete Process With Methods.

En la siguiente figura se muestra el Whith Method:

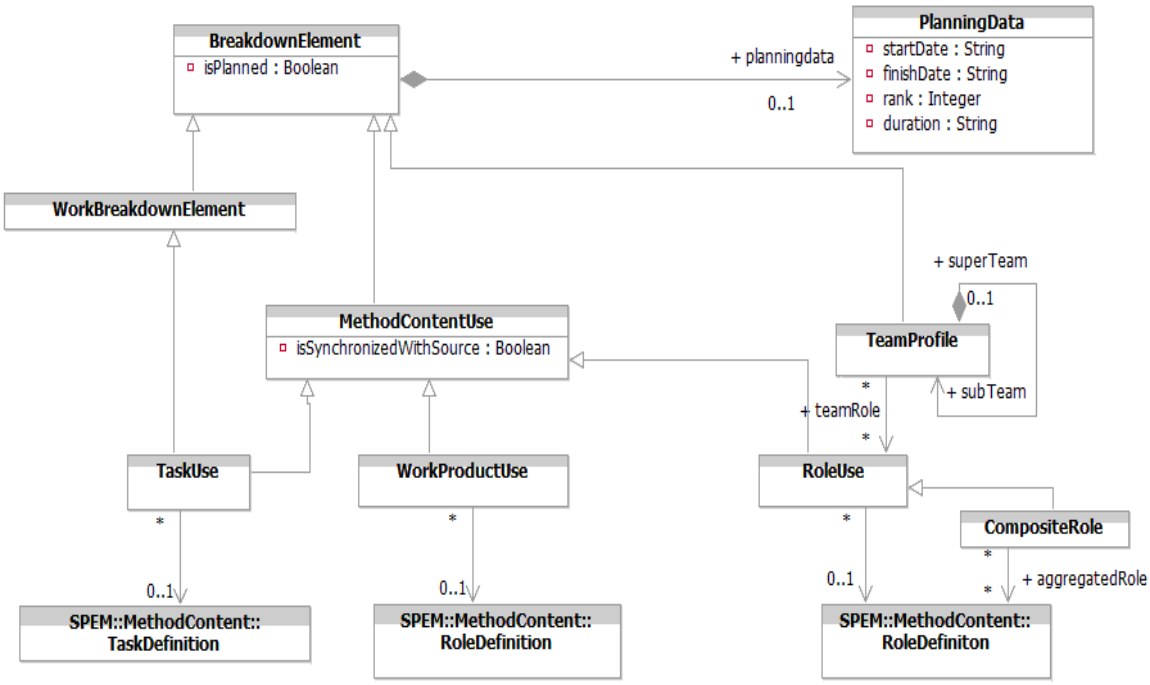

Ilustración 14. Paquete Process Whith Method. [VUL01]

Method Plug-In: Este paquete introduce los conceptos ("Method Plug-In", "Process Component", "Variability", etc.) para diseñar, gestionar y mantener repositorios y librerías de "Method Content" y "Process".

Permitiendo de este modo su reutilización, a través de los mecanismos de extensibilidad y variabilidad.

Algunos de los componentes más destacados de este paquete son:

COMPONENTE	**DESCRIPCION**
Actividad	Extensión de Activity para proporcionarle capacidades de variabilidad.
Method Configuration	Subconjunto lógico dentro de una Method Library definido mediante una selección de Method Packages.
Method Library	Contenedor de Method Plugins y definiciones de Method Configuration. Puede albergar cualquier

	elemento de SPEM 2.0.
Method Library Packageable Element	Cualquier elemento que puede contener una Method Library.
Method Plugin	Unidad de almacenamiento de la configuración, modularización, extensión, empaquetado e implantación de Process y Method Content.
Method PluginPackageable Element	Cualquier elemento que puede ser empaquetado dentro de un Method Plugin.
Process Component	Especialización de Process Package a la que se le pueden aplicar los conceptos de encapsulación y que contiene exactamente una Activity.
Process Component Use	Aplicación de un Process Component dentro de cualquier otro Process.
Section	Elemento que extiende Section (del paquete Managed Content) con capacidades de variabilidad.
Variability Element	Componente que proporciona capacidades de variación y extensión a los elementos de SPEM que derivan de él.
Variability Type	Enumeración de valores (contributes, replaces, extends, extendsreplace) para instancias de Variability Element.
WorkProduct Port	Entradas y salidas de Work Products para un Process Component.
Work Product Connector	Elemento utilizado para conectar Work Product Ports con el objetivo de

	ensamblar Process Components.

Ilustración 15. Componentes del paquete Method Plug-In.

En la siguiente figura se muestra el Method Plug-In

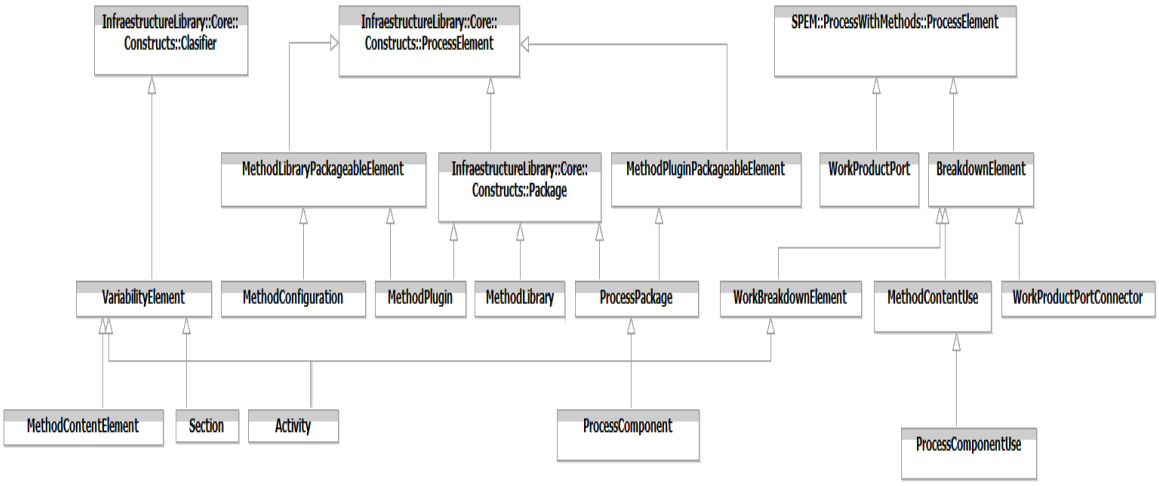

Ilustración 16. Method Plug-In. [VUL01]

Utilizando unos u otros paquetes, un ingeniero de procesos puede disponer de diferentes capacidades, conjuntos de conceptos y niveles de formalismo para expresar sus procesos. Los escenarios más habituales son los siguientes:

a) "Core" + "Managed Content" + "Method Content": En este caso no hay modelos de proceso formales definidos, sólo se define un repositorio para gestionar la documentación de descripciones de métodos de desarrollo y mantenimiento, técnicas y mejores prácticas.

b) Anterior + "Process Structure" + "Process Behavior": Aquí ya existe un proceso definido pero no son necesarios mecanismos avanzados para organizar o gestionar un repositorio de métodos, o para permitir diferentes vistas de un mismo proceso.

c) Los siete paquetes completos se emplean cuando se quiere disponer de toda la potencia y funcionalidad de SPEM 2.

3.1.4 Paquete SPEM Method Content

La idea básica de proceso en SPEM es presentada principalmente por el paquete "Method Content", por eso es importante detenerse en él para ampliarlo.

La idea central de SPEM 2 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 quién 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 quien (Rol) realiza qué (Tarea) para, a partir de unas

entradas (Productos de Trabajo) obtener unas salidas (Productos de Trabajo), en base o con la ayuda de algo (Guías).

Por esto, los cuatro tipos de elementos de contenido son: Tarea, Rol, Producto de Trabajo y Guía. También, existe un quinto tipo de elemento de contenido incluido con fines de clasificación y agrupación, llamado Categorías.

A continuación se describen sus características:

❖ **Tareas**

Una Tarea ("Task Definition") describe una unidad de trabajo asignable y gestionable, es decir, es la unidad atómica de trabajo para definir procesos.

Es un Elemento de Método que define el trabajo realizado por roles, pero también es una Definición de Trabajo (en procesos). Una Tarea está asociada con:

1..* Roles, distinguiendo entre:

- 1 realizador principal obligatorio [responsable]
- 0..* realizadores adicionales opcionales

1..* Productos de Trabajo como:

- Entradas obligatorias
- Entradas opcionales
- Salidas

0..* Herramientas, que se recomienda usar.

- 0..* Pasos, que describen de forma secuencial el trabajo a realizar.
- 0..* Habilidades, que se requieren habitualmente para llevar a cabo la tarea.

Cada Tarea tiene asociado, también, un "purpose" (objetivo); "key considerations" (factores clave); "alternatives" (alternativas); "steps" (lista de pasos), que detalla el trabajo a realizar de forma ordenada.

❖ **Roles**

Un Rol ("Role Definition") define un conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo. No se deben 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 varios individuos. Un rol es un Elemento de Método usado en las Definiciones de Tareas para señalar quienes las realizan. Un Rol está asociado con:

- 0..* Productos de Trabajo, de los que es responsable.
- 0..* Habilidades, que el rol típicamente provee.

Un Rol tiene asociado, también, "key considerations" (factores clave); "skills" (habilidades); "assignment approaches" (propuestas de asignación); "synonyms" (sinónimos).

❖ **Producto 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 asociaciones de los siguientes tipos:

- Composición (“Composition”), cuando las instancias de un Producto de Trabajo sirven para componer instancias de otro Producto de Trabajo. Ejemplo: “Actores” se emplean para componer “Casos de uso”.
- Agregación (“Aggregation”), si un Producto de Trabajo está formado por agregación de otros. Ejemplo: el “Manual de usuario” incluye el “Manual de instalación”.
- Es impactado por (“Impacte by”), cuando un Producto de Trabajo impacta en otro, es decir, si los cambios del primero obligan a cambiar el segundo. Ejemplo: si cambia el “Modelo de casos de uso”, es necesario adaptar a dicho cambio la “Realización de casos de uso”.

Existen tres tipos predefinidos de Productos de Trabajo:

Artefacto (“Artifact”): de naturaleza tangible (modelo, documento, código, archivos).

Un artefacto puede estar formado 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 (“stakeholder”). Está asociado con, cero o muchos, componentes entregables (“Deliverable Component”), que son los Productos de Trabajo, habitualmente artefactos, que lo forman.

Resultado (“Outcome”): un Producto de Trabajo de naturaleza intangible (resultado estado), o que no está formalmente definido.

Los Productos de Trabajo tienen asociado, también, “unique ID” (ID exclusivo); “purpose” (objetivo); “key considerations” (factores clave); “impact of not having” (indica el impacto de no tener este producto de trabajo dentro de un proceso); “reason for not needing” (motivos para no necesitarlo). Además, los “Artifacts” tienen “brief outline” (esquematización breve), y “representation options” (opciones de representación). Los “Deliverables” también tienen “external description” (descripción externa), “packaging guidance” (guía de empaquetado), y “deliverable parts” (lista de otros “Work Products” que forman parte del “Deliverable”).

❖ **Guías**

Una guía o instrucción (“Guidance”) es un elemento de método (o de proceso) que provee información adicional relacionada con otros elementos. Por ejemplo: ayuda o da información sobre cómo trabaja un rol, cómo crear un Producto de Trabajo, cómo usar una herramienta o cómo realizar una tarea. [GUIA]

SPEM 2 tiene predefinidos varios tipos de guías:

Activo Reutilizable (“Reusable Asset”): Provee una solución a un problema para un contexto dado. Incluye reglas o instrucciones sobre cómo utilizarlo.

Concepto (“Concept”): Resumen de ideas clave asociadas con principios básicos subyacentes.

Refieren a tópicos más generales que las directrices y abarcan varios productos de trabajo y/o actividades. Ejemplo: “iterativo e incremental”.

Definición de Término (“Term Definition”): Contiene definición de un término, concepto o idea relevante. Sirve para generar una especie de glosario automático. Se relacionan con Elementos de Contenido mediante su aparición en las descripciones textuales.

Directriz (“Guideline”): Provee detalle adicional sobre cómo realizar una tarea o grupo de tareas, o provee información adicional, reglas y recomendaciones sobre productos de trabajo. Entre otros, puede incluir detalles sobre: las mejores prácticas y aproximaciones diferentes para hacer un trabajo; cómo usar cierto tipo de Producto de Trabajo; los subtipos y variantes de un

artefacto y su evolución a lo largo del tiempo; habilidades que los roles deben adquirir o mejorar; medidas del progreso o madurez, etc.

Documentación ("Whitepaper"): Es una versión especial de Concepto que ha sido revisada o publicada externamente y que puede ser leída y comprendida de forma aislada.

Ejemplo ("Example"): Ejemplo de una instancia típica, parcialmente completada, de uno o más productos de trabajo o descripción del escenario en que una tarea debe ser realizada.

Guía de Herramienta ("Tool Mentor"): Explica el uso de una cierta herramienta en el contexto de cierto trabajo, o de forma independiente.

Guía para la Estimación ("Estimation Considerations"): Contiene indicaciones para estimar el esfuerzo asociado con cierto trabajo, incluyendo consideraciones sobre cómo hacer la estimación y las métricas a utilizar.

Hoja de Ruta ("Roadmap"): Hoja de ruta que describe, en forma de camino lineal, cómo suele llevarse a cabo una actividad o proceso complejos. Provee información sobre como las actividades y tareas se relacionan entre sí a lo largo del tiempo. Sólo pueden estar asociados a Actividades y Procesos.

Informe (“Report”): Plantilla predefinida de un resultado que se obtiene de forma automática mediante alguna herramienta.

Lista de Comprobación (“Checklist”): Identifica una serie de ítems (“check ítems” o elementos de comprobación), que deben ser completados o verificados.

Material de Soporte (“Supporting Material”): Comodín para utilizar cuando se está en un caso que no encaja en ninguno de los demás tipos de guías.

Plantilla (“Template”): Establece la tabla de contenidos, secciones, cabeceras y formato estandarizado predefinido de un artefacto (documentos, modelos). Puede incluir descripciones sobre cómo usar y completar cada parte.

Práctica (“Practice”): Manera o estrategia predefinida de hacer un trabajo que tiene un impacto positivo sobre la calidad de un Producto de Trabajo o de un proceso. Son ortogonales a los métodos y procesos, de forma que una práctica resume aspectos que pueden impactar en diferentes partes de un método o proceso. Ejemplos: “gestionar riesgos”, “verificación continua de la calidad”, “desarrollo centrado en la arquitectura”, “desarrollo basado en componentes”.

❖ **Categorías**

Una Categoría (“Category”) es un elemento de contenido, o de proceso, usado para categorizar, es decir, clasificar o agrupar dichos elementos en base a los criterios que desee el ingeniero de procesos. Una categoría puede tener, 0 o muchas, subcategorías. Esto permite establecer

cualquier tipo de jerarquía de agrupamiento de elementos. SPEM 2 distingue dos clases de categorías:

- Estándar ("Standard Category"): Vienen predefinidas en SPEM.
- Personalizada ("Custom Category"): Sirven para que el ingeniero de procesos pueda definir otras categorías nuevas.

En SPEM 2 se incluyen 5 tipos predefinidos de categorías (Categoría personalizada):

- Conjunto de Roles ("Rol Set"): Sirven para agrupar roles que tienen algo en común (usan técnicas similares, requieren habilidades parecidas). Ejemplo: Analista englobando Analista de Sistemas e Ingeniero de Requisitos.

- Disciplina ("Discipline"): Permiten categorizar el trabajo (tareas). Una disciplina es una colección de tareas que están relacionadas con un área principal de esfuerzo dentro de un proyecto completo. Suelen estar basadas en una perspectiva tradicional de proyectos en cascada: requisitos, análisis, diseño, construcción, pruebas, mantenimiento, gestión del proyecto, aseguramiento de calidad, etc.

- Dominio ("Domain"): Permiten establecer una jerarquía de dominios, para clasificar productos de trabajo, con tantos niveles como se desee. El nivel inferior son productos de trabajo y el resto de los niveles son dominios y subdominios. Al ser una jerarquía, un Producto de Trabajo sólo puede estar asociado con un único dominio. Ejemplos: "modelo", "código".

- Herramienta (“Tool”): Sirve para categorizar guías de herramientas. Por tanto, para asociar herramientas con tareas, roles o productos de trabajo deberá emplearse una categoría personalizada.

- 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”. [GUIA]

Cada clase de “Content Element” tiene un icono predefinido por el perfil UML de SPEM, que lo caracteriza en los diagramas y documentación utilizados o generados también por las diversas herramientas (como EPF Composer). Con fines de personalización, a cada tipo de “Work Product” y “Guidance” se le puede asociar otro icono específico (como se ve en la descripción anterior).

Asociaciones de Content Elements

Entre los distintos tipos de “Content Elements” pueden existir asociaciones, que permiten representar las relaciones entre estos, ellas son:

- Task-Steps: lista ordenada de pasos que se llevan a cabo en una tarea.
- Task-Roles: “Primary Performer” (realizador principal, obligatorio); “Additional Performers” (realizadores adicionales).
- Task-Work Products: “Mandatory Inputs” (entradas obligatorias); “Optional Inputs” (entradas opcionales); “Outputs” (salidas).
- Task-Guidances: “Guidances” relacionadas (sólo tipos “Checklist”, “Concept”, “Estimating Guideline”, “Example”, “Guideline”, “Reusable Asset”, “Supporting Material”, y “Tool Mentor”).
- Task-Categories: “Disciplines” y “Custom Categories” a las que pertenece la “Task”.
- Role-Work Products: “Responsible For” (responsable de); “Work Products that are Output of Tasks that this Role Perfoms” (productos de trabajo que son salida de tareas que realiza el rol).
- Role-Guidances: “Guidances” relacionadas (sólo tipos “Checklist”, “Concept”, “Example”, “Guideline”, “Reusable Asset”, y “Supporting Material”).
- Role-Categories: “Role Sets” y “Custom Categories” a las que pertenece el “Role”.
- Work Product-Guidances: “Guidances” relacionadas (sólo tipos “Checklist”, “Concept”, “Estimating Guideline”, “Example”, “Guideline”, “Report”, “Reusable Asset”, “Supporting Material”, “Template”, y “Tool Mentor”).

- Work Product-Categories: “Domains”, “Work Product Kinds” y “Custom Categories” a las que pertenece el “Work Product”.
- Guidance-Guidances: lista de “Guidances” (de tipo “Checklist”, “Concept”, “Example”, “Guideline”, “Reusable Asset” y “Supporting Material”) que pertenecen al “Guidance” principal. Todos los tipos de “Guidance” pueden incluir otros “Guidance”, salvo el tipo “Practice” que en lugar de incluir otros “Guidances” está relacionado con diversos “Content Elements”.

Otras relaciones secundarias que se pueden representar en SPEM 2 de manera directa o indirecta, según la herramienta editor utilizada (EPF Composer no tiene opciones directas para ello) son:

- Task – Tools: herramientas que se utilizan para hacer una Tarea.
- Tool- Work Products: Productos de Trabajo que se producen utilizando una herramienta.
- Rol – Qualifications: Habilidades que posee un Rol.
- Task – Qualifications: Habilidades requeridas para realizar una Tarea.
- Work Product – Work Products: Un Producto de Trabajo (source) es necesario para poder producir otro Producto de Trabajo (target).

3.1.5 Procesos

Para trabajar en SPEM y definir un proceso, como primer paso se deben crear los componentes del “Content Method” y como segundo paso combinar estos componentes en actividades y procesos.

El contenido de los procesos se puede especificar desde varias vistas: [GUIA]

• **Estructura de desglose de trabajo:** En esta vista se trabaja con las tareas que componen las actividades, iteraciones, fases, etc.

• **Asignación de equipos:** Esta vista muestra los roles que participan en cada actividad. También se pueden definir directamente desde esta vista los roles que participan en la actividad, para después definir los productos y las tareas de las que son responsables.

• **Utilización del producto de trabajo:** Esta vista muestra los productos de entrada y salida de cada actividad. En esta vista se pueden definir directamente los productos de salida de la actividad, para después definir las tareas en las que se generan esos productos.

Creación de proceso:

Existen dos formas de abordar la especificación de un Proceso:

- Descendente: El proceso se desarrolla definiendo la jerarquía de actividades y luego se indican las tareas que la forman, las actividades del nivel más bajo en la jerarquía.

- Ascendente: Primero se crean las actividades con las tareas en Patrones de Procesos, luego se definen patrones más complejos, componiendo otros patrones creados. Por ultimo se define el proceso componiendo patrones complejos.

Definiciones sobre Procesos:

Una Definición de Trabajo (“Work Definition”) es un concepto abstracto de SPEM, describe el trabajo que se ejecuta en un Proceso. Este trabajo puede ser “Activity”, “Phase”, “Iteration” y “LifeCycle”.

Una Definición de Trabajo puede estar asociada con, 0 o muchas, precondiciones (restricciones que deben cumplirse para que el trabajo pueda comenzar) y con, 0 o muchas, poscondiciones (restricciones que deben cumplirse para que el trabajo pueda considerarse concluido).

Un Elemento de Desglose (“Breakdown Element”) es una generalización abstracta para cualquier tipo de elemento que aparece en un Proceso y es parte de una estructura de desglose. Los Elementos de Desglose de Trabajo (“Work Breakdown Element”) son el principal tipo de elemento de desglose, ya que representan la descomposición del trabajo. [UNI01]

Existen dos tipos de elementos de desglose: Actividad e Hito (llamado Objetivo en EPF).

Las propiedades comunes a ambos son:

- Se Puede Repetir (“Is Repeatable”): puede haber varias iteraciones o repeticiones.
- Continuo (“Is Ongoing”): es un trabajo sin duración fija o estado final. Ejemplo: trabajo de un gestor de proyecto que 1 hora al día se dedica a revisar el estado de avance de las tareas.
- Condicionado por Sucesos (“Is Event Driven”): su inicio no está determinado por eventos normales (cuando acaba el trabajo que lo precede o cuando se concluye algún Producto de Trabajo), sino por otro evento especial.

Aporta tres propiedades importantes:

- Admite Varias Apariciones (“Has Multiple Ocurrences”): al realizar el proceso puede haber más de una instancia del elemento.
- Es Opcional (“Is Optional”): no es obligatoria su inclusión cuando se lleva a cabo el proyecto.
- Planeado (“Is Planned”): El elemento es incluido al generar los planes de proyecto que se exportan a las herramientas de gestión de proyectos.

Los tipos de Elementos de Desglose son:

- Parámetros de Procesos, para asociar productos de entrada/salida.
- Realizadores de Procesos, para asociar roles.
- Secuencias de Trabajo, para las relaciones de precedencia.
- Elementos de Desglose de Trabajo.
- Roles en Uso.
- Productos de Trabajo en Uso.
- Relaciones entre Productos de Trabajo en Uso.

- Asignaciones de Responsabilidad en Procesos.

Cada elemento de desglose tiene asociada una estructura de desglose de trabajo ("WBS, Work Breakdown Structure") que representa su estructura interna y, opcionalmente, un flujo de trabajo ("Work Flow").

En SPEM 2 la jerarquía de desglose del trabajo, es decir, los conceptos existentes para representar el esfuerzo a realizar a distintos niveles de detalle, son los siguientes del más general al más particular.

- **Delivery Process** (Proceso de Despliegue): Representa un proceso tan complejo como se necesite, que será el que sirva de base para realizar cierto tipo de proyectos. Describe una aproximación completa e integrada, abarcando un ciclo de vida completo de desarrollo o mantenimiento. Sirven como plantillas para planificar y ejecutar los proyectos. En un Proceso de Despliegue se ensamblan Patrones de Proceso y Elementos en Uso (Tareas, Roles y Productos de Trabajo en uso). Ejemplos: "proceso unificado de Rational (RUP)", "programación extrema (XP)", "métrica 3", etc.

- **Capability Pattern** (Patrón de Capacidad): representa un patrón de proceso, es decir, un fragmento de proceso que puede ser reutilizado más de una vez en un "Delivery Process". Describe un grupo de actividades reutilizables como solución a algún tipo de problema o situación habitual. Se definen para poder ser empleados más de una vez en uno o varios procesos o con fines de organización. Se pueden almacenar en una jerarquía de Paquetes de Procesos ("Process Package").Algunos escenarios de uso de patrones de procesos son:

  - Servir como bloques para construir Procesos para Despliegue o Patrones de Proceso más complejos.
  - Ayudar a la ejecución de proyectos que no siguen un proceso bien definido, sino que trabajan en base a fragmentos de proceso (buenas prácticas) de una manera flexible (métodos ágiles).
  - En formación, para describir el conocimiento de una cierta área clave, buena práctica, disciplina, etc.

- **Activity** (Actividad): es el elemento central para definir procesos ya que permite organizar los elementos básicos de proceso (Roles, Productos de Trabajo y Tareas). Representa una unidad de trabajo general en un Proceso. Una actividad puede tener estructura interna formada por agregación de Elementos de Desglose, que pueden ser de varios tipos, no sólo de trabajo: actividades más simples (anidamiento), elementos de método en uso (Roles en uso, Productos de Trabajo en uso), e Hitos. [GUIA].Una actividad representa una unidad de trabajo general asignable a realizadores específicos, representados por Roles en Uso. Dicha asignación se hace a través de Realizadores de Proceso ("Process Performer"). Una Actividad puede tener entradas y producir salidas,

representadas por Productos de Trabajo en Uso. Para esta última asociación se utilizan los llamados Parámetros de Proceso ("Process Parameter").

- **Task** (Tarea): es la porción de trabajo más pequeña en el modelo de procesos en SPEM.

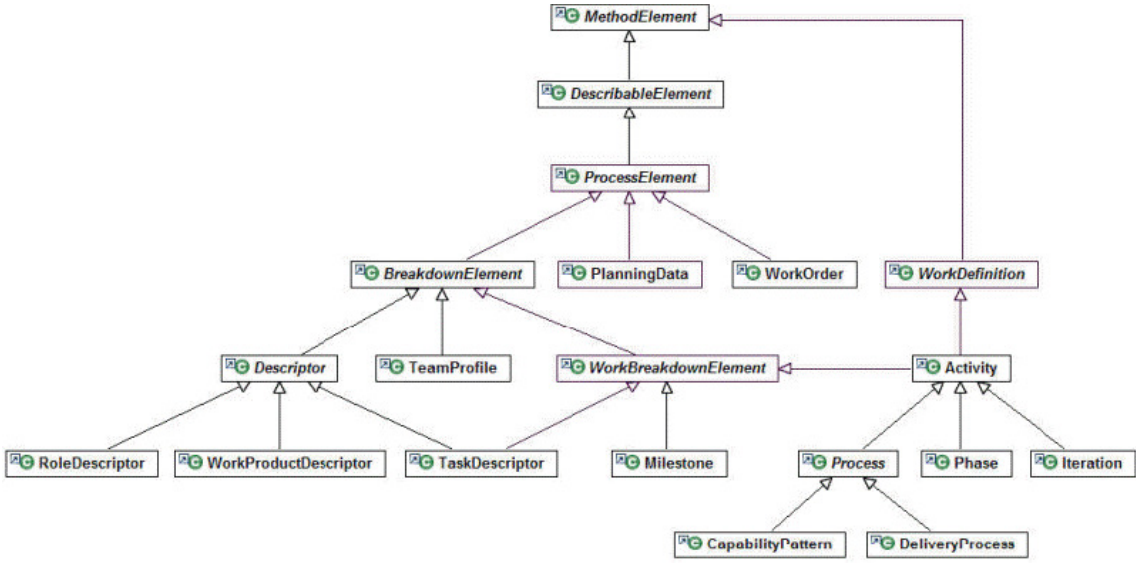

Ilustración 17. Jerarquía de conceptos de Procesos. [GUIA]

Actividad en SPEM:

SPEM 2 tiene predefinidos varios tipos especiales de Actividades: Iteración, Fase y Proceso. Puesto que Actividad es un concepto genérico, que incluye a proceso como especialización, en SPEM 2 no se respeta la nomenclatura tradicional de los estándares ISO de Ingeniería de Software. En especial, debe tenerse en cuenta que SPEM no sigue los tres niveles estáticos de descomposición utilizados en el modelo de procesos de ISO/IEC 12207: procesos formados por actividades formadas por tareas.

La actividad es la principal subclase de "Work Definition". Ésta describe una parte del trabajo desarrollado por un "Process Role": las tareas, operaciones y acciones que son desempeñadas por un Rol o las que el Rol puede asistir. Una Actividad se puede componer de elementos atómicos llamados pasos.

Fases, Iteraciones e Hitos (Tipos especiales de actividades):

Las fases e iteraciones son dos tipos especiales de actividades (adicionales a los procesos), los hitos son un tipo de elemento de desglose de trabajo, distinto de los otros dos, actividades y tareas. [GUIA]

Una Fase (Phase) representa un período de tiempo que es significativo para un proyecto, y que acaba con un punto de control de gestión importante, un hito o un conjunto de entregables concluidos. En la práctica, en SPEM 2 una fase es una actividad que tiene el valor falso en la propiedad “Es repetible”.

Por el contrario, una Iteración (Iteration) representa un conjunto de actividades anidadas que se repiten más de una vez. Sirve para organizar ciclos repetitivos de trabajo. En la práctica, en SPEM 2 una iteración es una actividad que tiene el valor verdadero en la propiedad “Es repetible”.

Un Hito (Milestone) representa un evento significativo para el desarrollo de un proyecto:

- Una decisión importante,

- La conclusión de un entregable,

- La conclusión de una fase, etc.

3.1.6 Elementos en uso

Los elementos del contenido de método (Tareas, Roles y Productos de Trabajo) reutilizados en los procesos reciben el nombre de elementos en uso. Un elemento en uso es una instancia de un elemento de método particularizada para un contexto de proceso determinado. Por ello, los elementos en uso no son reutilizables (para reuso ya están las descripciones de los elementos en el contenido de método).

Así, una Tarea en Uso (“Task Use”) representa la ocurrencia de una Definición de Tarea (Contenido de Método) en el contexto de una Actividad. En ella se pueden precisar y modificar, respecto de la tarea original, su documentación, pasos, roles y productos de entrada y de salida. [GUIA]. Además, se pueden definir dos asociaciones que no aparecen en las tareas como elementos de contenido de método: los roles que asisten en la tarea y las entradas externas.

Existen dos maneras de utilizar tareas en actividades:

a) Reutilizar una descripción de tarea del contenido de método, que pasa a ser una tarea en uso, o

b) Crear directamente en la actividad una instancia de tarea en uso. En este caso no se tendrá descripción asociada, sino sólo el nombre.

Un Producto de Trabajo en Uso (Work Product Use) representa la ocurrencia de un Producto de Trabajo real en el contexto de una Actividad, pudiendo modificar su documentación. Además, se pueden definir dos atributos que no aparecen en la definición de Producto de Trabajo en el contenido de método: los estados de entrada y de salida del producto en la actividad. Por ejemplo, un producto "Requisitos" puede entrar en una actividad en el estado "identificados" y salir en el estado "refinados". También existen dos maneras de utilizar Productos de Trabajo en actividades equivalentes a las dos indicadas para las Tareas:

a) Reutilizar una descripción de producto de trabajo del contenido de método, que pasa a ser un producto de trabajo en uso.

b) Crear directamente en la actividad una instancia de Producto de Trabajo en uso, que sólo tendrá nombre pero no descripción asociada.

De forma similar a las dos anteriores, un Rol en Uso (Role Use) representa la ocurrencia de un Rol real en el contexto de una Actividad, pudiendo particularizar la documentación, productos de los que es responsable o que modifica, y equipos (roles compuestos) a los que pertenece.

También existen dos maneras de utilizar roles en actividades:

a) Reutilizar una descripción de rol del contenido de método, que pasa a ser un rol en uso. SPEM le llama Realizador ("Performer").

b) Crear directamente en la actividad una instancia de rol en uso. En este caso no se tendrá descripción asociada, sino sólo el nombre. SPEM le llama Participante ("Participant").

Un Rol Compuesto ("Composite Role") es un rol en uso especial, que se corresponde con más de una Definición de Rol del Contenido de Método.

Un proceso se puede ver como una colaboración entre roles para alcanzar cierta meta u objetivo.

3.1.7 Reutilización y variabilidad de Elementos de Contenido y Procesos

La organización en Plugins ("Plug-in") permite que se puedan reutilizar los elementos de contenido y los procesos definidos en una Biblioteca ("Library"). Dicha reutilización se puede realizar de dos maneras:

a) Al crear un Plugin nuevo se puede referenciar a otros Plugins.

b) Usar de forma directa el contenido de un Plugin desde otro Plugin diferente.

Para reutilizar el contenido de un Plugin con algunas modificaciones existe el mecanismo de variabilidad ("Variability"), que permite modificar elementos de método o de proceso (adiciones, cambios, omisiones) sin modificar directamente el original. Dichas diferencias afectan a las propiedades, es decir, a los atributos y a las asociaciones con otros elementos. [GUIA]

SPEM 2.0 contempla 5 tipos de variabilidad entre elementos de contenido:

- **No Asignado**
- **Contribuye**
- **Amplía,**
- **Reemplaza**
- **Amplía y Reemplaza**

No asignado ("not assigned"): No existe relación de variabilidad entre el elemento en cuestión y otros. Es el valor por defecto.

Contribuye ("contributes"): Un elemento contribuye a otro elemento base, si añade sus valores de atributos e instancias de asociación a éste, sin modificar directamente las propiedades que ya tenia dicho elemento base.

Reemplaza ("replaces"): Mediante este tipo de variabilidad un elemento puede reemplazar (sustituir) los atributos y las asociaciones de salida de un elemento base, sin modificar directamente ninguna propiedad de éste. El elemento que reemplaza se incluirá en las vistas generadas mientras que el reemplazado no aparecerá. Un elemento base sólo puede ser reemplazado por otro único elemento en una misma configuración. Si se define más de un reemplazo para un mismo elemento base, entonces no se realizará ningún reemplazo.

Amplía ("extends-herencia"): En este caso, el elemento hereda las propiedades del elemento base que amplía, dejando totalmente intacto al elemento base. El elemento incluye las propiedades heredadas del elemento base más las propias (por ejemplo, nuevas asociaciones). Ambos elementos aparecen en las vistas generadas.

Amplía y Reemplaza ("extends-replaces"): Este tipo de variabilidad combina los efectos de las variabilidades de tipo Amplía y de tipo Reemplaza, ya comentadas. El resultado es que las propiedades que no se han definido en el elemento nuevo que amplía y reemplaza se heredan del elemento base. En cambio, las nuevas propiedades definidas sustituyen a las del elemento base. El elemento que amplía y reemplaza aparece en las vistas que se generan, pero el elemento base no. Este tipo de variabilidad se utiliza al generar Plugins que renombran elementos, reemplazan descripciones, etc., sin remodelar completamente todas las relaciones y atributos del Plugin base.

Cuando se resuelven varias variabilidades, las reglas de prioridad son las siguientes:

- Primero se realizan las asociaciones de variabilidad de tipo "Contribuye", seguidas de las de tipo "Reemplaza", luego de las de tipo "Amplía" y, por último, las de tipo "Amplía y Reemplaza". - Entre dos variabilidades del mismo tipo, primero se realiza la que está más arriba en la jerarquía.

La variabilidad está definida para variar métodos, no procesos, variando el método del que se genera el proceso, de todas formas se estaría variando el proceso.

Una desventaja de la variabilidad es que los elementos para introducir variabilidad son muy complejos y por esto, difíciles de reutilizar, y tampoco existe una notación específica que ayude en el proceso de variabilidad.

3.1.8 Configuraciones de métodos

Una Configuración de Método ("Method Configuration") es una selección de contenidos de los Plugins de una Biblioteca de Métodos ("Method Library").

Este mecanismo de SPEM permite tener distintas vistas de una misma biblioteca de Plugins o de un único Plugin.

La definición de una configuración puede estar basada en las definiciones de otras configuraciones. Si alguna de estas configuraciones base cambian, los cambios serán automáticamente válidos también en la nueva configuración, reduciendo de esta manera el esfuerzo de mantenimiento.

Una configuración de método define un subconjunto lógico de una biblioteca de métodos mediante la selección (filtro) de los paquetes deseados, tanto de contenido de método como de procesos. La selección de elementos incluidos en una configuración se puede refinar añadiendo o sustrayendo de la configuración todos los elementos asociados con una cierta categoría. [GUIA]

Un repositorio o biblioteca de métodos y procesos ("Method Library") en SPEM 2 es una colección de uno o más Plugins y una o varias configuraciones ("Configuration"). Cada Plugin se almacena en un directorio de disco diferente e incluye contenido de método ("Method Content") y procesos ("Processes").

A su vez, el contenido de método está formado por paquetes de contenido ("Content Package"), categorías estándar ("Standard Category") y categorías personalizadas ("Custom Category"); y el apartado de Procesos contiene patrones de proceso ("Capability Pattern") y procesos para despliegue ("Delivery Process").

 Biblioteca de métodos

Hasta aquí un resumen que intentó describir las principales características del metamodelo SPEM. Como se intentó exponer, el mismo presenta muchas ventajas que justifican por qué elegirlo para trabajar en Procesos de Software.

Gracias a él se pueden definir procesos a partir de la combinación de partes existentes en el modelo, utilizar el lenguaje natural reconociendo la existencia de partes que no pueden formalizarse, y brindar la posibilidad de crear un repositorio con el que contar para almacenar los procesos que serán recuperados mas tarde.

Además permite tener una representación estandarizada y una gestión de librerías de métodos reutilizable.

Como así también soportar el desarrollo, gestión y el crecimiento de procesos, basándose en la reutilización de fragmentos de método.

Estos fragmentos se crean independientemente del ciclo de vida del proceso, pudiendo ser personalizados dependiendo del contexto de desarrollo de cada proceso. Ofrece además el uso de configuraciones que permite ver la definición de los procesos y componentes relevantes para cada entorno, desplegando sólo la parte que se necesita en cierto contexto.

Desde un punto de vista de mercado presenta la ventaja de contar con la existencia de una herramienta gratuita, permitiendo una utilización más rápida por las organizaciones, ya que no tienen que contar con el costo de la propia herramienta.

3.2 Eclipse Process Framework Composer

Eclipse Process Framework Composer (EPF Composer) es una herramienta para quienes se dedican a mantener e implementar Procesos de Software.

Facilita el trabajo de ingenieros de procesos brindándoles la posibilidad de definir y elaborar Procesos de Software, evitando los problemas que surgen en las organizaciones, cuando todos los miembros deben entender los métodos y prácticas, las actividades que se definen, las tareas que las componen. También en las organizaciones se necesita un acceso de la información que sea común a todos los miembros y que los mismos cuenten con prácticas comunes y regulares para logar un mejor desempeño.

Además los equipos de desarrollo deben definir cómo se aplican sus métodos de desarrollo y mejores prácticas a través del ciclo de vida de un proyecto, es decir, ellos deben definir o seleccionar un Proceso de Software, necesitando para esto una herramienta que les permita ver claramente las relaciones entre las actividades de los procesos, sus entradas, sus salidas y los actores que las ejecutan con sus roles claramente definidos.

De esta manera, se lograría poder establecer de forma sencilla las prioridades según el proyecto específico, e incluso determinar las dependencias entre actividades para tener en cuenta que pasos se deben seguir para lograr el objetivo deseado, así como también tener el acceso a guías de trabajo que ayuden a realizarlo.

EPFC brinda la posibilidad de intentar mejorar estas problemáticas, ofreciendo un diseño de administrador de contenido que provee una estructura de administración y aspectos 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, los cuales poseen su propia forma y formato.

EPF Composer tiene catálogos de procesos predefinidos para situaciones particulares, que pueden ser adaptados a necesidades individuales.

Como principio fundamental de EPF Composer, se define la posibilidad de la separación del contenido reusable del método, de su aplicación en los procesos. [EPF01]

El proyecto de Eclipse Process Framework tiene como objetivos:

- Brindar la posibilidad de acceder de forma rápida, fácil y centralizada a la información.
- Permitir contar con contenidos de los Procesos de Software en formatos estándar que ofrezcan una fácil integración.
- Soportar la definición de procesos y estandarización de prácticas y procesos dentro de las organizaciones.

3.2.1 Presentación del entorno

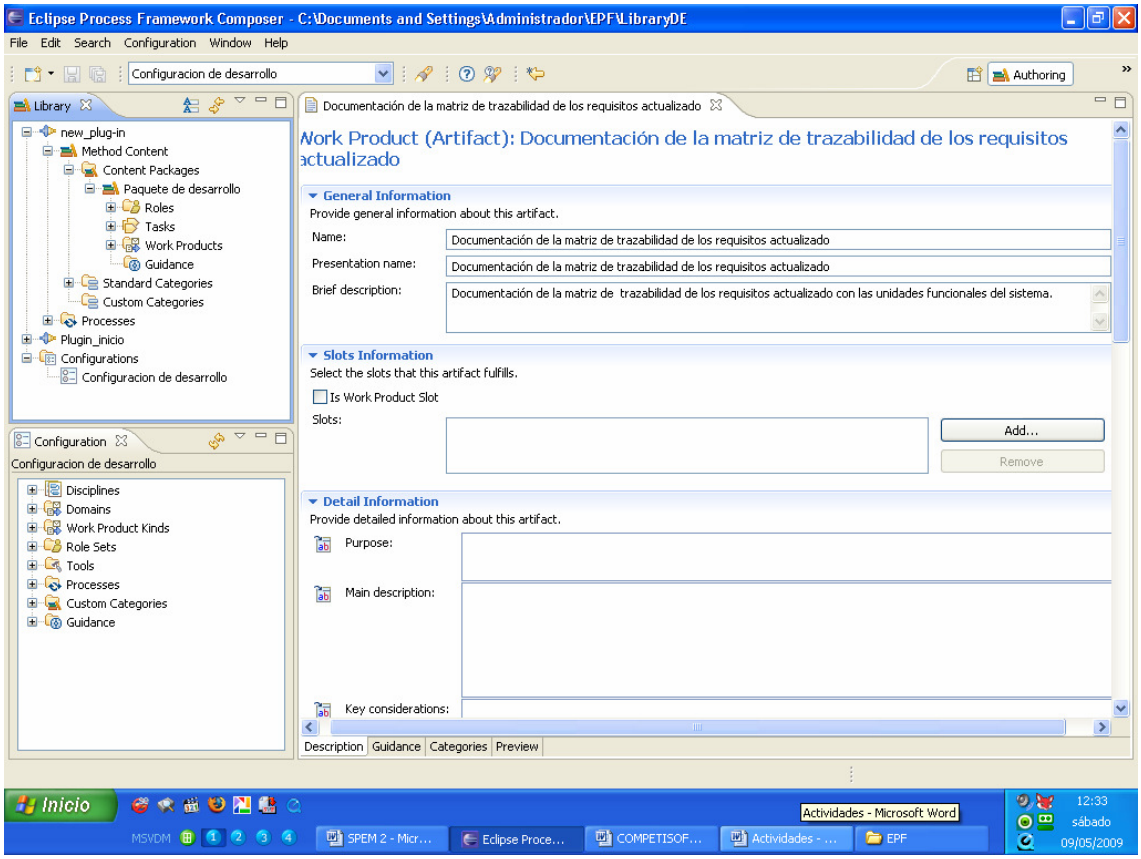

Ilustración 18. Entorno EPF.

3.2.2 Arquitectura de la herramienta

EPF Composer es una aplicación Java construida sobre varios componentes de código abierto.

Para instalar la herramienta se debe contar con Java Runtime Environment (JRE) 1.4.2 o superior, luego de instalar JRE, se requiere tener la herramienta, que se encuentra disponible para descargar en la página Web del proyecto EPF de Eclipse (http://www.eclipse.org/epf/downloads/tool/tool_downloads.php).

Se necesita descomprimir la descarga y ejecutar epf.exe que se encuentra en la carpeta resultante al descomprimir el archivo que se descarga. [EPF03]

3.2.3 Escenarios de uso

Para empezar a utilizar la herramienta es necesario conocer que existen diferentes escenarios de uso, por ejemplo, se puede:

- Seleccionar y configurar contenidos de método y procesos existentes. De esta forma cada proyecto puede seleccionar los procesos y contenidos de métodos que se ajusten a las necesidades propias, a través de las librerías de métodos, que permiten reunir los contenidos de método. Luego de ser adquiridos estos, se configura un proceso, seleccionando o quitando paquetes de método, según se consideren o no necesarios en el proyecto particular.

- Personalizar un proceso existente. Aquí se modifican procesos existentes usando el editor de EPFC, así se adapta a las necesidades del proyecto particular, a diferencia del método anterior que se configuraban los procesos existentes para decidir cuales mostrar y cuales no. Es decir, se puede directamente adicionar, remover o reemplazar elementos en el proceso. Por lo tanto, en contraste con cambiar la configuración, lo cual hace cambios globales en el proceso, se pueden definir cambios individuales sólo en los lugares requeridos.

- Crear un nuevo proceso. Con esta posibilidad que brinda la herramienta se pueden crear procesos desde cero completamente nuevos, en el cual se reutilicen actividades de uno o más procesos existentes, o bien creando todos sus componentes por primera vez.

- Desarrollar contenido de método y crear o extender procesos. Este último escenario permite no sólo crear o adecuar procesos reutilizando el contenido del método de EPF o de terceros, sino desarrollar contenido del método propio y usarlo en la adecuación de procesos existentes o la creación de nuevos procesos.[EPF01]

3.2.4 Editor EPF Composer

Eclipse Process Framework Composer, llamado EPFC, es una herramienta gratuita, desarrollada dentro del entorno ECLIPSE, que sirve para editar fragmentos de método, procesos o metodologías, y generar automáticamente la documentación adecuada en formato para la Web.

Dichos fragmentos se almacenan en formato XMI y, al estar basados en el estándar SPEM 2, pueden ser reutilizados por cada vez más herramientas CASE. En realidad, EPFC utiliza la "Unified Method Architecture" (UMA), que a su vez está basada en el metamodelo SPEM 2.

EPFC es un editor de procesos SPEM 2, que incluye opciones adicionales para publicar de forma automática sitios Web. [EPF03]

Brinda la posibilidad de trabajar en la herramienta con dos perspectivas diferentes, es decir presenta dos vistas, cada una de las cuales posee un conjunto de funciones disponibles para tareas específicas. Estas dos perspectivas de trabajo son “Authoring”, como una perspectiva de edición y “Browsing”, como una perspectiva de navegación.

Authoring

La perspectiva Authoring o de edición, provee de vistas y funciones para navegar y editar o crear contenido de métodos y procesos.

La perspectiva Authoring provee dos vistas en paneles separados: La vista de librería y la vista de configuración. Al hacer doble click sobre cualquier elemento de alguna de esas vistas, se abre el panel de edición en la derecha. El panel de edición contiene varios tabs a través de los cuales se puede editar información sobre el elemento que se ha seleccionado.

Para seleccionar la perspectiva de Authoring, se usa el menú “Abrir perspectiva” en la barra de tareas principal y se selecciona Authoring.

Browsing

La perspectiva Browsing (Exploración) permite hacer una vista previa y navegar a través de una configuración del método sin hacer ningún cambio.

La perspectiva Browsing contiene la vista de configuración, que muestra el contenido en la configuración seleccionada. Al hacer click sobre cualquier elemento en el panel de configuración, se puede hacer una vista previa del elemento en la vista de contenido, mostrando como dicho elemento aparecerá publicado en un sitio Web. La vista de contenido provee funcionalidades de navegación similares a las de un Explorador o Browser.

Se debe hacer click sobre cualquier vínculo para ir a la página referenciada. Usando los botones en la barra de herramientas de la vista de contenido, se pueden realizar funciones similares a las de un Browser, por ejemplo, ir atrás o refrescar.

Para seleccionar la perspectiva Browsing, se hace click sobre el botón "Abrir Perspectiva" y se selecciona Browsing. [EPF03]

En la siguiente imagen se muestra el uso de estas perspectivas de trabajo:

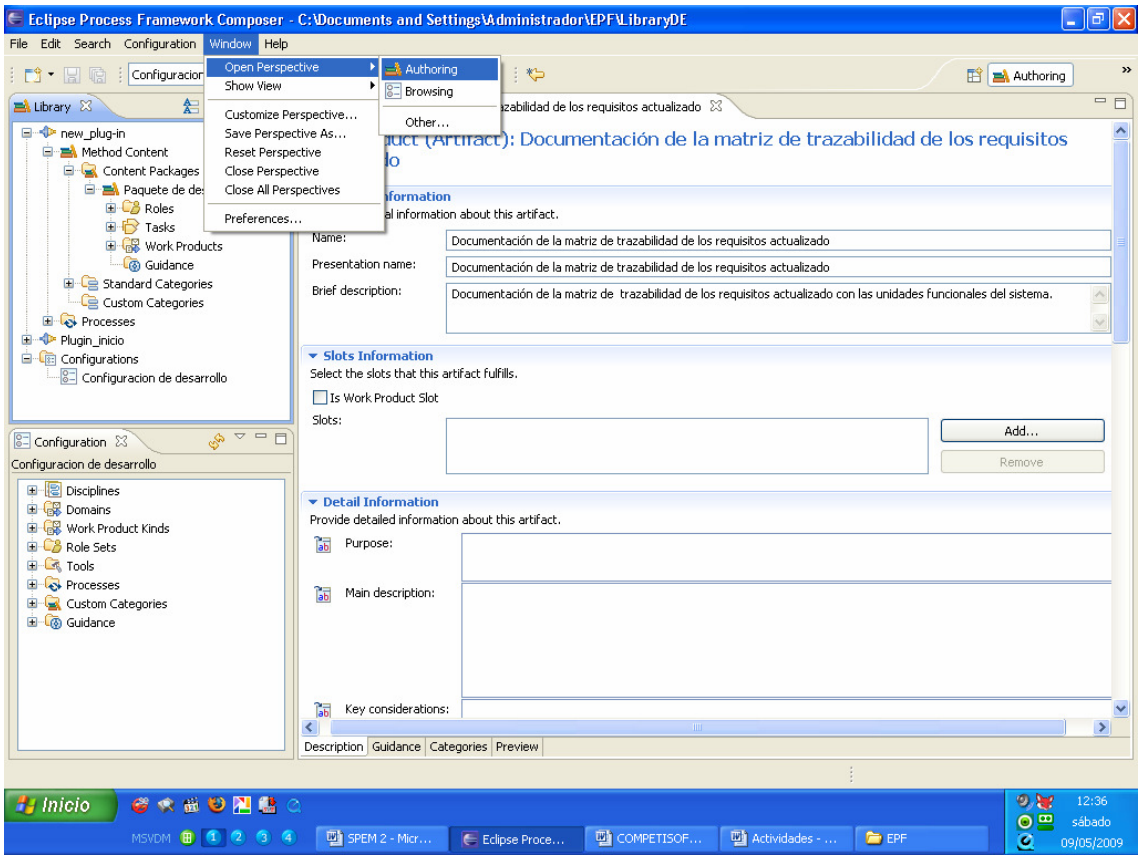

Ilustración 19. Perspectivas de trabajo.

3.2.5 Composición de procesos (Pasos para su creación)

El objetivo primordial de la composición de procesos, es dividir el contenido en una parte general y reusable, que puede ser usada a través de todos los procesos de forma general y otra más específica, y aplicada al proyecto particular que se está desarrollando.

Existen diferentes formas de crear un proceso, dependiendo del escenario que se elija utilizar o de la manera que se quisiera trabajar, es decir de forma ascendente o descendente, como se explicó en la descripción del metamodelo SPEM.

Una de las formas más ventajosas de hacerlo en cuanto a mantenibilidad, reusabilidad y escalabilidad es la siguiente:

En primer lugar se debería crear una **Biblioteca de métodos,** donde poner todos los Plugins y configuraciones de la metodología a definir.

La **Biblioteca de métodos** es un contenedor físico de plug-ins de método y definiciones de configuración de métodos. Todos los elementos del método están almacenados en una biblioteca de métodos. Los plug-ins de métodos están compuestos por contenido de método ("Method Content") y procesos ("Process"), el contenido del método contiene paquetes de contenido y categorías, que pueden ser estándares o personalizadas, mientras que los procesos estructuran este contenido en fragmentos de proceso llamados patrones de capacidad ("Capability Patterns") y procesos de ciclos de vida completos llamados procesos de entrega ("Delivery Processes"). Conceptos que se describieron en la sección del metamodelo SPEM.

Una biblioteca de método también tiene una o más configuraciones de método, que filtran la biblioteca y proveen de conjuntos de trabajo más pequeños de contenido de la biblioteca para el usuario final.

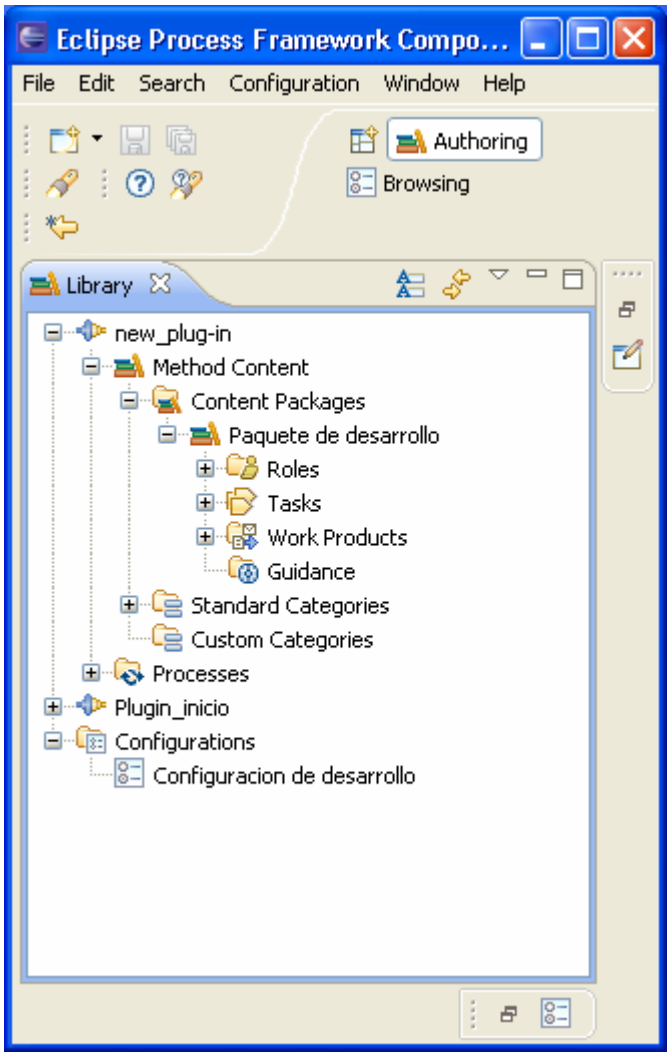

Ilustración 20. Librería de métodos.

Luego es aconsejable crear un **Plugin de Métodos.** El uso de Plugins, facilita la estructuración, organización y modularización.

Un Plugin contendrá los elementos necesarios para la definición de un proceso (o varios si tienen una relación estrecha).

El siguiente paso es crear **Elementos de métodos**, estos serán utilizados posiblemente luego para crear los procesos, aumentando así la reutilización y mantenibilidad del proceso, estos se reutilizarán tantas veces como se necesite y si es necesario hacer cambios, estos se harán sólo en la definición del elemento del método y no en cada ocurrencia del elemento del proceso. [GUIA]

Además estos elementos de contenido sirven como bibliotecas de conocimiento, muy importante en las organizaciones que desean tener esta información al alcance de todos sus desarrolladores, de forma accesible y organizada, gracias a la posibilidad de poder definir paquetes, estos elementos se pueden organizar eficientemente. [EPF02]

El EPF define a los actores (roles) como aquellos que poseen una serie de habilidades, competencias y responsabilidades dentro de un equipo de desarrollo. En particular elaboran una serie de Productos de Trabajo ("Work Products"), mediante la ejecución de una serie de tareas ("Tasks"). Basándose en el concepto de SPEM.

Esto permite representar la situación que aparece en las organizaciones desarrolladoras de software donde se presenta un proyecto que es llevado a cabo por un equipo de personas donde cada una debe tener su rol bien definido y sus tareas bien asignadas y programadas.

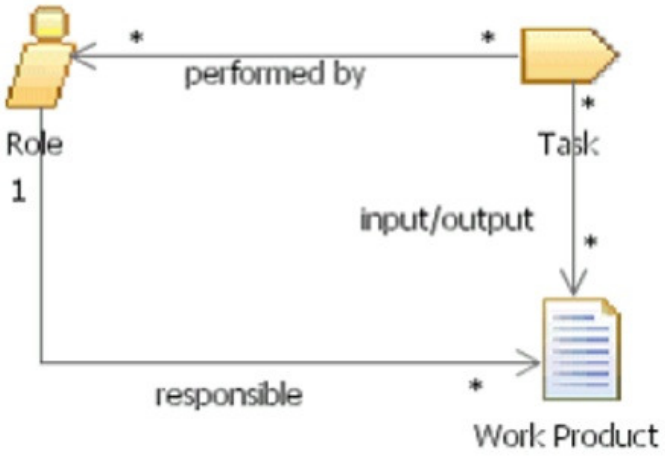

Ilustración 21. Contenido de métodos. [EPF03]

La creación de elementos y de paquetes de contenido con EPFC se hace en la vista "**Biblioteca**".

La especificación de los atributos (descripción breve, objetivos, etc.) se realiza en la pestaña "**Descripción**" del editor del elemento. Cada tipo de elemento cuenta con una serie de pestañas mediante las que se pueden definir sus asociaciones con otros tipos de elementos. Así, las tareas tienen una pestaña en la que se definen los roles que participan en ellas, otra pestaña en la que se definen los productos de entrada y de salidas; los roles tienen una pestaña en la que se definen los productos de los que son responsables, etc.

En cada paquete de contenido se crean de forma automática cuatro carpetas para almacenar las definiciones de cada tipo de elemento.

Como se muestra a continuación:

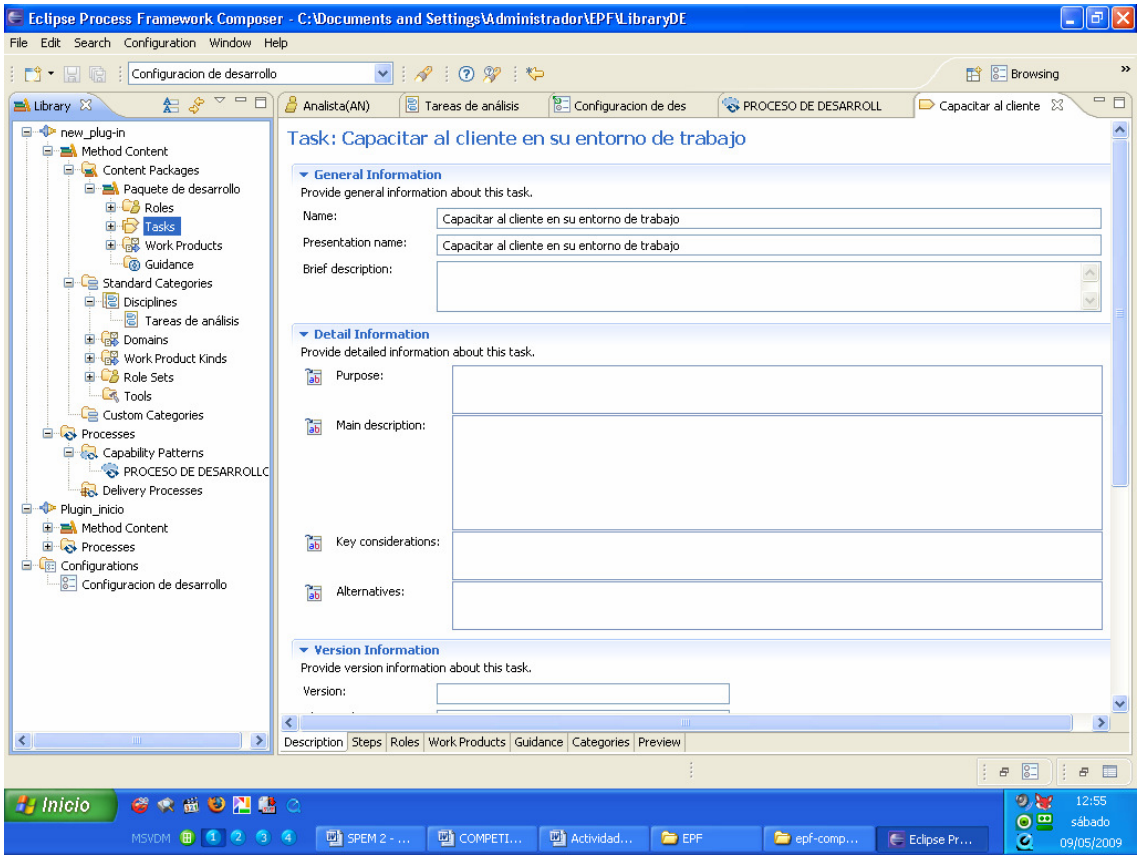

Ilustración 22. Contenido del Paquete.

El orden en la creación de los elementos de método no es relevante, aunque por comodidad se aconseja seguir el siguiente orden:

1) **Crear las guías**, ya que pueden ser referenciadas desde cualquier otro elemento de método.

La información asociada a la guía en EPF es:

Name: Nombre interno con el que es reconocido la guía.

Presentation name: Nombre con el que la guía será publicada en los archivos HTML.

Type: Tipo de guía.

Brief description: Descripción corta sobre la guía.

Main description: Descripción principal y detallada sobre la guía.

Version: Versión actual de la descripción da la guía.

Variability Type: Relaciones de la guía con otras guías.

Check Items: Items de la guía.

Guiance: Guías con la que se asocia.

Preview: Vista previa de la guía.

2) **Crear los productos de trabajo**, al crear cada producto, se deberá asociarle las guías convenientes.

La información asociada a un Producto de Trabajo dentro del EPF es la siguiente:

Name: Nombre interno con el que es reconocido el producto de trabajo.

Presentation name: Nombre con el que el producto de trabajo será publicado en los archivos HTML.

Brief description: Descripción corta sobre el producto de trabajo.

Purpose: Propósito de la existencia del producto de trabajo.

Main description: Descripción principal y detallada sobre el producto de trabajo.

Key considerations: Consideraciones clave que se deben tener en cuenta para el producto de trabajo.

Brief outline: Descripción breve de la forma en que debe ser entregado el producto de trabajo.

Representation options: Opciones para la presentación del producto de trabajo a la hora de ser entregado.

Impact of not having: Información sobre el impacto de no tener este producto de trabajo dentro de un proceso.

Reason for not needing: Razón o razones por las cuales este producto de trabajo puede no ser necesario dentro de un proceso.

Version: Versión actual de la descripción del producto de trabajo.

Change Date: Última fecha de actualizaciones o cambios efectuados sobre la descripción del producto de trabajo.

Change Descriptions: Descripción de los cambios realizados entre versiones.

Authors: Nombre de las personas que crearon o actualizaron el producto de trabajo.

Copyright: Información sobre los derechos de propiedad intelectual de los autores sobre el producto de trabajo.

Content Variability: Relaciones del producto de trabajo con otros productos de trabajo.

Icon: Personalización de los íconos para el producto de trabajo.

Guidance: Guías que están asociadas al producto de trabajo.

Categories: Categorías a las cuales pertenece este producto de trabajo.

Preview: Vista previa de cómo se verá el producto de trabajo cuando se publique en un sitio Web.

3) **Crear los roles**, al crear cada rol, se debe indicar los productos de trabajo de los que son responsables y asociarle las guías convenientes.

La información asociada a un Rol dentro del EPF es la siguiente:

Name: Nombre interno con el que se identifica el rol.

Presentation name: Nombre que aparece en los archivos HTML que genera EPF Composer.

Brief description: Descripción corta y sin detalles del rol.

Main description: Descripción detallada del rol.

Key Considerations: Consideraciones clave que se deben tomar con respecto al rol.

Skills: Habilidades que debe tener la persona para desempeñar el rol.

Assignment Approaches: Forma en que se debe escoger la persona para que desempeñe el rol.

Synonyms: Otros nombres con los que se conoce el mismo rol.

Version Information: Información sobre la versión actual y cambios que ha sufrido la tarea.

Content Variability: Relaciones del rol con otros roles.

Work Products: En esta vista se puede editar la relación que tiene el rol con los productos de trabajo asociados.

Guidance: Vínculos para información adicional sobre el rol en forma de guías.

Categories: Categorías a las cuales pertenece este rol.

Preview: Vista previa de cómo se verá el rol cuando se publique en un sitio Web.

4) **Crear las tareas**, al crear cada tarea, se deberá indicar los roles que participan en la tarea, los productos de trabajo que son entrada o salida de la tarea y las guías convenientes.

La información asociada a una tarea dentro del EPF es la siguiente:

Name: Nombre que define la tarea.

Brief description: Pequeña descripción o explicación de la tarea.

Domain: Conjunto de tareas a los que corresponde la misma.

Purpose: Resultado específico planeado y deseado hacia el cual está dirigida la tarea.

Main description: Explicación más desarrollada de la tarea.

Steps: Conjunto de todos los pasos que se llevan a cabo en una tarea, representa todos los pasos, que supone como consecuencia, el objetivo de la tarea.

Key considerations: Consideraciones claves a tener en cuenta para la ejecución de la tarea.

Alternatives: Alternativas posibles que se disponen, posiblemente menos efectivas, para llevar a cabo en la tarea.

Version Information: Información sobre la versión actual y cambios que ha sufrido la tarea.

Roles: Roles encargados de realizar la tarea.

Work Products: Productos de trabajo de entrada y salida de la tarea.

Guidance: Guías de la tarea.

Categories: Categoría a la que pertenece la tarea.

Preview: Vista previa de la tarea.

More info: Conjunto de elementos que ayudan a una descripción más eficiente de la tarea, como elementos que proporcionan detalles adicionales de cómo realizar una tarea específica, ideas auxiliares asociadas con principios básicos, etc.

La ventaja de respetar este orden es que cuando se crea un elemento ya existen todos los elementos con los que está relacionado y, por tanto, su edición es más cómoda. Las descripciones de los elementos de método y sus relaciones con otros elementos se indican en las distintas pestañas de la vista de edición de cada elemento. [GUIA]

Luego se pueden definir las categorías estándar para la **Categorización de Elementos de Método**.

Los elementos del contenido de método se pueden categorizar mediante su asociación con categorías estándar (disciplinas, dominios, clases de productos de trabajo, conjuntos de roles y herramientas), para lo cual existe una pestaña en EPFC.

Además, los elementos del contenido de método (y también los elementos en uso en procesos) se pueden organizar alternativamente mediante categorías personalizadas. Las categorías personalizadas también se crean en EPFC a través de la pestaña correspondiente en la vista "Biblioteca". Estas categorías personalizadas pueden contener cualquier tipo de elemento, proceso, categoría, etc. Son útiles para organizar de forma lógica elementos que no se puedan organizar mediante las categorías estándar, como por ejemplo, agrupar todas las plantillas de productos, agrupar todas las guías que sean directrices, etc.

Las categorías personalizadas permiten organizar el contenido de acuerdo al esquema que se desee, sirviendo como medio eficaz para organizar el contenido de una publicación.

El contenido de una categoría personalizada se especifica en la pestaña "**Asignar**" del editor de la categoría. [EPF03]

Existen diferentes tipos de categorías estándar: disciplinas para categorizar tareas, dominios y clases de productos de trabajo para categorizar productos de trabajo, conjuntos de roles para categorizar roles y herramientas para categorizar guías de herramienta (un tipo específico de guía) de una herramienta específica.

Además, las disciplinas y los conjuntos de roles se pueden agrupar mediante agrupaciones

de disciplinas y agrupamientos de conjuntos de roles, respectivamente.

La asignación de elementos a una categoría se hace desde una pestaña específica de la vista de edición de la categoría.

A continuación, y antes de crear un proceso, se debe crear una **Configuración**.

Esto es necesario porque cada elemento de proceso debe especificar su configuración (o configuraciones, ya que un proceso puede estar asociado con varias configuraciones). Las configuraciones son subconjuntos de la biblioteca de métodos que permiten restringir la vista de todos los elementos de la biblioteca a únicamente el conjunto de elementos que interese.

La selección de elementos de la configuración se hace desde la pestaña "Selección de paquete y plug-in" de la vista de edición de la configuración. [GUIA]

Una configuración de un método se conforma de tres partes:

- Una descripción de la configuración.
- Una selección del conjunto de paquetes.

- Una selección de vistas que serán publicadas en el sitio Web.

Las vistas que se asocian a la configuración, se refieren generalmente a elementos que aparecen bajo el tipo de "Categorías" dentro de EPF Composer. Hay que tener en cuenta que cuando se publica contenido, lo que se publica es una configuración. Por tanto, si se quiere publicar varias vistas habrá que hacerlo por separado, dando como resultado varios sitios Web distintos.

Para entregar versiones reducidas a terceros también se utilizan las configuraciones y la capacidad de exportarlas que ofrece EPF. Así, se incluirán en la configuración todos los elementos que deban entregarse a terceros. Se exportará la configuración y posteriormente se importará en una nueva biblioteca de métodos vacía que únicamente contendrá la configuración exportada y todos los elementos seleccionados en dicha configuración. De esta forma, se puede entregar a terceros una biblioteca de métodos que únicamente contiene los elementos que se desean entregar. Hay que tener en cuenta que las referencias a los elementos que no se incluyen en la configuración exportada se pierden, y por tanto, hay que tener cuidado ante posibles pérdidas de información.

Las configuraciones ofrecen una alternativa en la creación de procesos, ya que puede definirse un proceso general, que incluya contenidos para varios tipos de procesos más específicos. Mediante el uso de configuraciones se seleccionan los contenidos de cada uno de esos tipos de procesos más específicos. Crear los diversos tipos de proyecto es sencillo seleccionando la configuración adecuada para el proceso.

Al seleccionar los elementos de la configuración, EPF Composer avisa si hay elementos referenciados que no se han incluido en la configuración. Se consideran dos tipos de advertencia, errores o avisos, en función del tipo de relación con el elemento ausente.

- Si el elemento ausente es un elemento referenciado por otro elemento que sí pertenece a la configuración (por ejemplo, un producto utilizado en una tarea incluida en la configuración), se indicará con un error.
- Si el elemento ausente es un elemento base para otro elemento que tiene relación de variabilidad amplía, contribuye o reemplaza con dicho elemento ausente, se indicará un aviso.

Dichos avisos y errores los puede solucionar automáticamente EPF calculando el cierre de todos los elementos referenciados e incluyéndolos en la configuración. Ya que el cierre es recursivo, se pueden añadir a la configuración elementos que tal vez no quieran añadirse. La otra opción para solucionar los avisos o errores es seleccionar manualmente los elementos ausentes que se consideren imprescindibles. Hay que tener en cuenta que la existencia de avisos y errores no impide la publicación de la configuración. Lo que ocurrirá es que los elementos ausentes no se publicarán y no aparecerán enlaces a los elementos ausentes referenciados.

En una configuración de método, además de seleccionar los contenidos de la configuración, se pueden seleccionar las vistas de la configuración. Las vistas son, básicamente categorías. Con los elementos definidos en la vista se generarán árboles de navegación mediante los que explorar el contenido de la configuración. Este concepto no aparece en el estándar SPEM, es un añadido de la herramienta EPF Composer.

Las vistas de una configuración se definen en la pestaña "**Vistas**" del editor de la configuración. A una configuración se le pueden añadir tantas vistas como se desee. El contenido de las vistas son categorías, por lo que se pueden elegir como vistas las categorías estándar, en las que cada categoría sería una vista, o, preferiblemente, utilizar categorías personalizadas. Esta segunda opción es más potente, ya que en una categoría personalizada se pueden incluir todos los elementos de todos los tipos que se desee. Hay que tener en cuenta que los elementos contenidos en la categoría que se elegirá como vista deben estar incluidos en la configuración, ya que en caso contrario, no se publicarán. [GUIA]

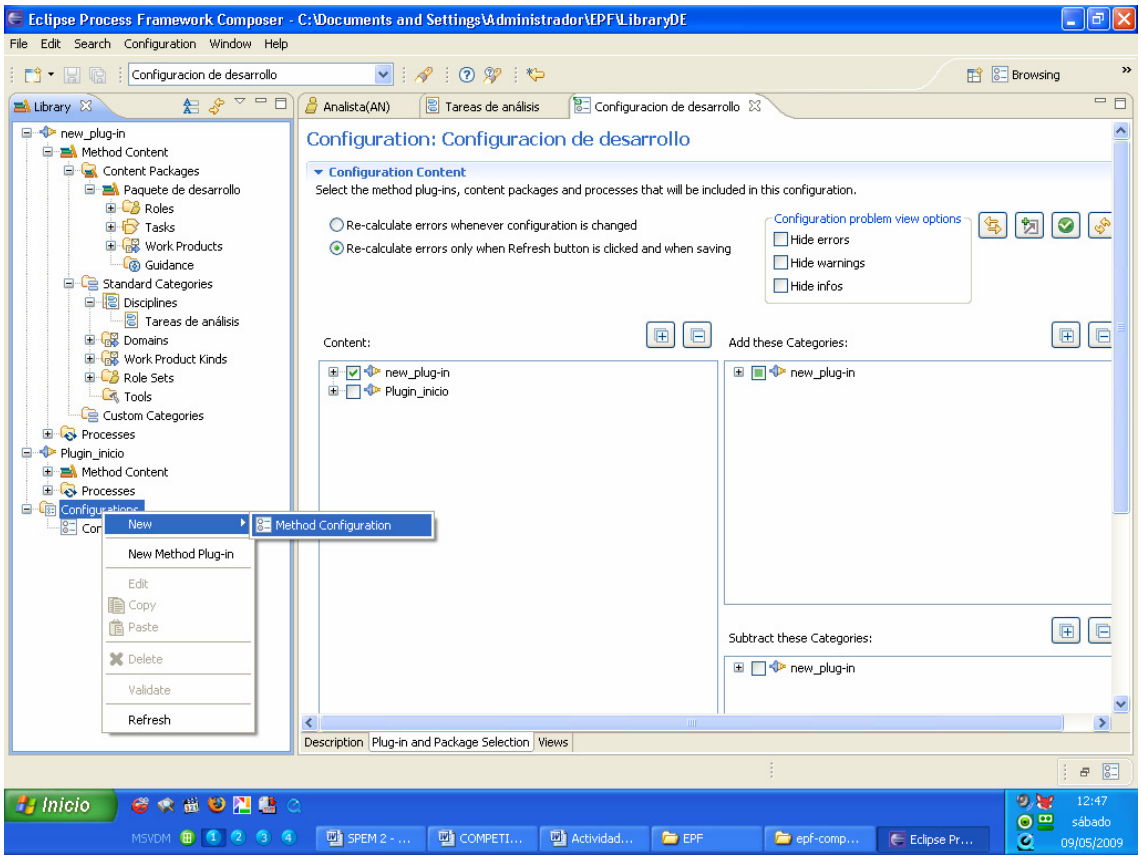

Ilustración 23. Configuración en EPF.

Una vez que se ha creado al menos una configuración, se pueden empezar a crear los **Patrones de Procesos**.

A la hora de crear procesos se recomienda utilizar un enfoque ascendente, ya que facilita la reutilización de los procesos definidos.

Con EPFC se pueden crear los dos tipos de procesos de SPEM 2:

- Patrón de proceso: Patrones de proceso de SPEM, que describen una agrupación reutilizable de tareas o actividades.
- Proceso para despliegue: Describen un enfoque completo e integrado para una metodología completa, un tipo de ciclo de vida, un tipo específico de proyecto, etc.

Al igual que los Elementos de Contenido, los Patrones de Proceso y los Procesos para Despliegue se pueden organizar y jerarquizar mediante paquetes de proceso.

La creación de Procesos y de Paquetes de Proceso se hace en la vista "Biblioteca".

Para aplicar dicho enfoque ascendente, lo primero es crear los patrones de proceso básicos, en los que se reutilizán las tareas definidas en el contenido de método.
La descripción y la estructura del proceso se definen en las distintas pestañas de la vista de edición del proceso.

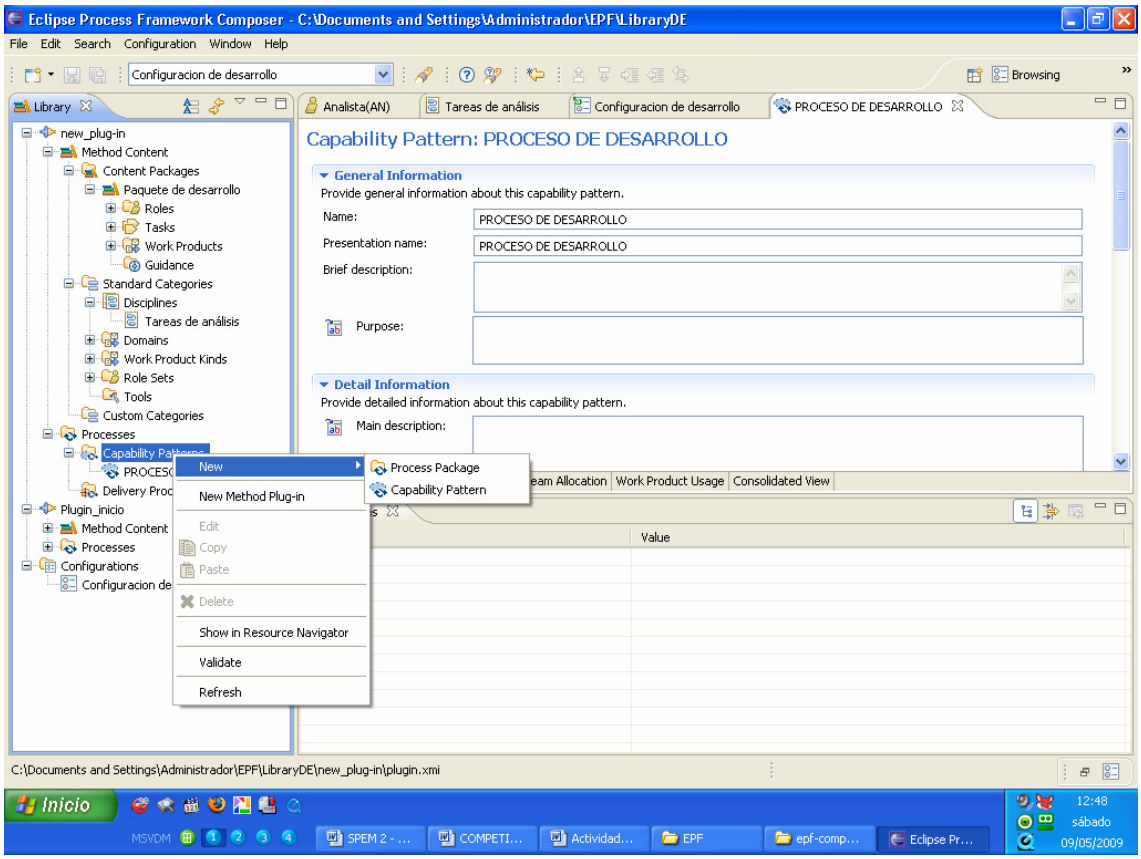

Ilustración 24. Proceso en EPF.

Para representar los dos tipos de procesos antes comentados (es decir, las llamadas actividades de forma general en SPEM 2), se utilizan estructuras de descomposición, mediante las cuales se representa la jerarquía de actividades (incluyendo fases e iteraciones) e hitos. Además, permiten ordenar la secuencia del trabajo (flujo de trabajo), indicando relaciones de precedencia entre los elementos de la estructura de descomposición.

Para ver y editar desde diferentes puntos de vista, la estructura de desglose de un proceso/metodología completo o de un fragmento, EPFC incluye las siguientes pestañas:

a) **Estructura de Desglose de Trabajo** (“Work Breakdown Structure”): En EPFC es una tabla que contiene la descomposición jerárquica de los elementos que definen el trabajo a realizar: actividades (patrones de proceso, fases, iteraciones), hitos, tareas, etc.

b) **Asignación de Equipos** (“Team Allocation”): Muestra una jerarquía similar a la anterior, pero mostrando para cada actividad (cada nivel) la lista de roles que participan.

c) **Utilización de Productos de Trabajo** (“Work Product Usage”): Esta perspectiva se centra en los productos de trabajo de forma similar a como la anterior se centra en los roles.

d) **Vista Consolidada** (“Consolidated View”): Combina las informaciones que se muestran en las tres vistas anteriores. Sólo existe con fines informativos, es decir, no permite edición.

Para desarrollar la estructura de desglose de un proceso o metodología se deben crear los elementos de desglose que lo forman: actividades (y sus variantes, fases e iteraciones), hitos (también llamados objetivos) y, en el nivel más bajo de la jerarquía, tareas, roles o productos de trabajo. Estos últimos dependerán de la perspectiva de desglose (pestaña) en la que se esté trabajando.

Los elementos de desglose se pueden crear directamente en la estructura de desglose o se pueden reutilizar del contenido de método (tareas, roles y productos) o de otros patrones de proceso (actividades). Las actividades e hitos se pueden añadir en cualquiera de las estructuras de desglose.

A la hora de especificar un proceso con EPFC hay que tener en cuenta que esta herramienta no impone restricciones en la jerarquía sobre los tipos de actividad. Por ejemplo, se puede incluir una fase dentro de una iteración, lo cual no tiene sentido. Por ello, es responsabilidad del ingeniero de procesos definir una jerarquía que sea lógica. [GUIA]

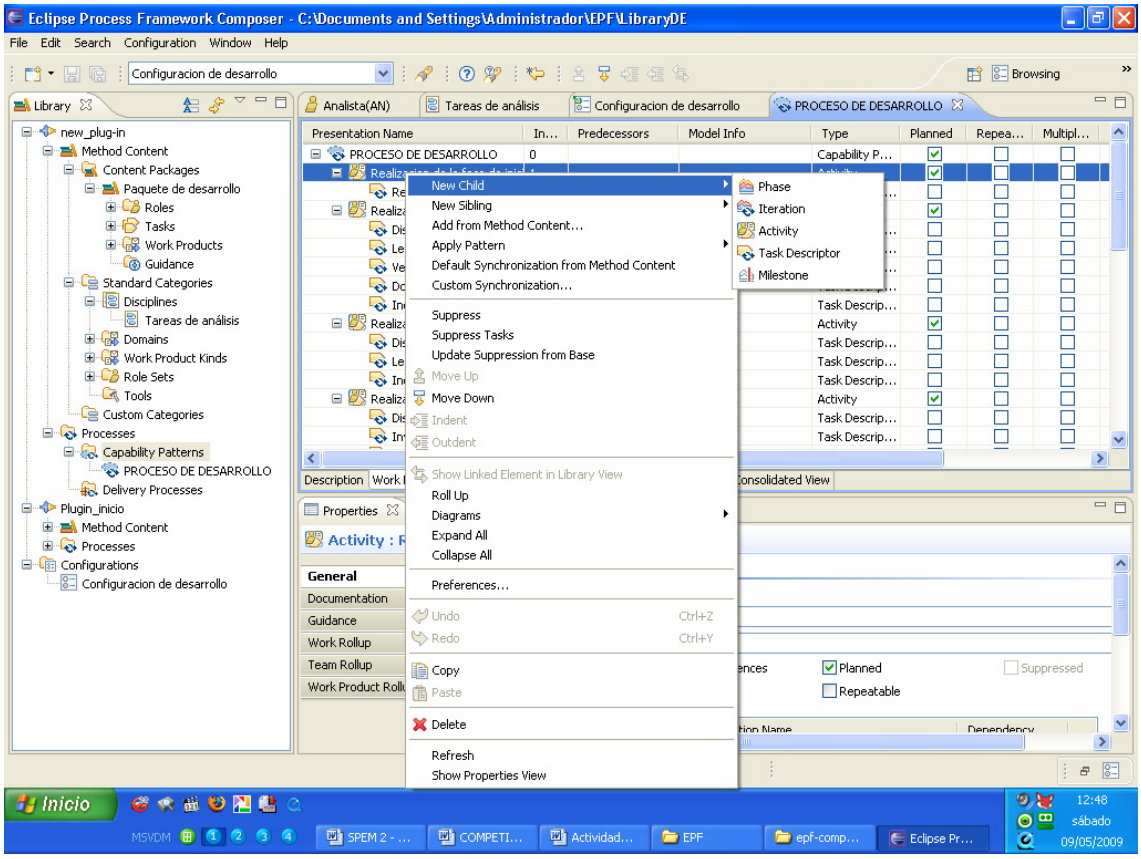

Ilustración 25. Actividad en EPF.

La perspectiva más útil para definir un proceso es el WBS, ya que en ella se establece la jerarquía de todos los esfuerzos o trabajos que deben ser realizados; utilizando las otras dos perspectivas de edición para los detalles de los participantes y de los productos asociados con cada Elemento de Desglose de Trabajo ("Work Breakdown Element") o WBE.

En el metamodelo SPEM 2 a partir del elemento raíz del WBS (un Patrón de Proceso o un Proceso para Despliegue), se añaden los WBE, que pueden ser de los siguientes tipos:

- Actividad ("activity"): Es el bloque básico para construir el desglose de los procesos. Una actividad representa un grupo de elementos de desglose (otras actividades, tareas, etc.).
- Descriptor de Tarea ("Task Descriptor"): Corresponde a la Tarea en Uso en SPEM 2. Es la reutilización en el WBS de una tarea. Cada vez que a un WBS se añade como elemento de desglose un elemento de método de tipo tarea, todos los roles y productos de trabajo asociados con ella se incorporan automáticamente al WBS.
- Hito (milestone): Describe un evento significativo para un proyecto, como por ejemplo, una decisión importante o haber completado una fase.

En el nivel más bajo de la jerarquía de un WBS, las actividades están compuestas por tareas.

Hay dos formas de añadir una tarea a una actividad: una es reutilizar las tareas definidas en el contenido de método y la otra es definir las tareas directamente en el proceso creando un nuevo hijo de tipo descriptor de tarea.

EPFC ofrece dos opciones para reutilizar tareas del contenido de método:

- Arrastrar desde la vista "Configuración".
- Hacer clic derecho en la actividad padre y seleccionar "Añadir desde el contenido de método".

Cada vez que se incluye una tarea en un proceso, lo que realmente se incluye es un descriptor de tarea (concepto de SPEM 2.0: "tarea en uso"). Lo mismo ocurre con los roles y productos que aparecen en un proceso: realmente, son descriptores.

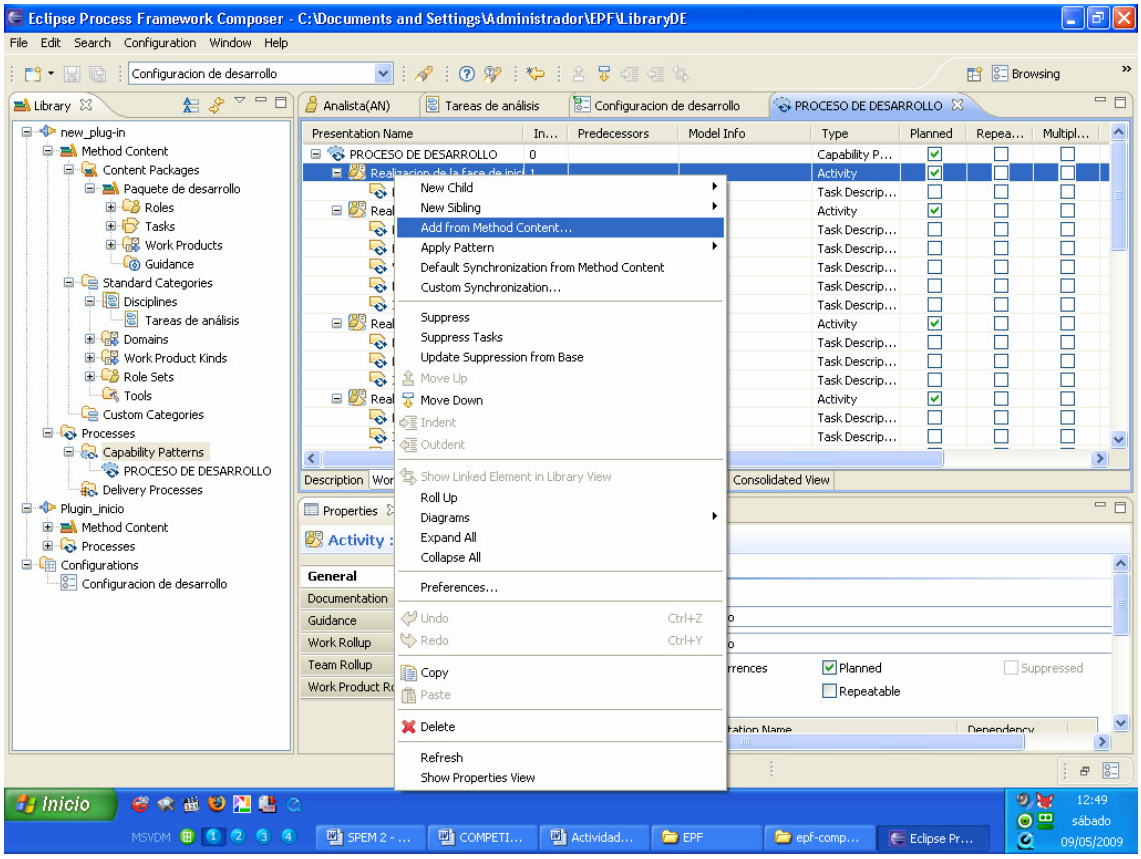

Ilustración 26. Tarea en EPF.

Descriptores de tareas, de roles y de productos de trabajo:

Estos conceptos en SPEM 2.0 reciben el nombre de "tarea en uso", "rol en uso" y "producto de trabajo en uso". El motivo por el que se utilizan los descriptores es que estos se pueden editar y personalizar para cada aparición del elemento y adaptarlos a las necesidades en el contexto del proceso en el que aparece dicho elemento.

De esta forma, se pueden modificar los atributos y las asociaciones del descriptor para reflejar la utilización concreta del elemento en el proceso sin modificar el elemento de método.

Las modificaciones a los descriptores se realizan en las distintas pestañas de la vista "Propiedades".

- Descriptor de rol: En un descriptor de rol se puede modificar su documentación, los productos de los que es responsable y los equipos a los que pertenece. No incluye ningún atributo o asociación diferente a los que aparecen en las definiciones de rol del contenido de método.

- Descriptor de producto: En un descriptor de producto se puede modificar su documentación. Además, se pueden definir dos atributos que no aparecen en la definición de producto de trabajo en el contenido de método: los estados de entrada y de salida del producto en la actividad. Por ejemplo, un producto "Requisitos" puede entrar en una actividad en el estado "identificados" y salir en el estado "refinados".

Los cambios realizados en un elemento de método no se propagan automáticamente a los descriptores que lo refieren. Para que un descriptor refleje los cambios realizados en un elemento de método después de aplicarlo en un proceso, se debe sincronizar el descriptor con el elemento de método. En la sincronización se perderán las modificaciones realizadas en el descriptor, ya que se copiarán los valores de los atributos y las relaciones del elemento de método. La sincronización de un descriptor se realiza haciendo click derecho en el elemento de la estructura (sea la de desglose de trabajo, de asignación de equipos o de utilización de productos) que se desee sincronizar y al pulsar "Sincronización predeterminada desde el contenido de método" se podrá sincronizar todos los atributos y relaciones, o al pulsar "Personalizar sincronización" se podrá elegir los atributos y relaciones que se quieran sincronizar y no perder los cambios realizados en el descriptor que interesa mantener.

También se pueden añadir descriptores de tarea, rol o producto de trabajo a una actividad que no refieran a ningún elemento de método. Para añadir un descriptor a una actividad se hace click derecho en la actividad a la que se quiere añadir, se selecciona "Nuevo hijo", y se añade el descriptor, que, dependiendo de la estructura en la que se esté trabajando, será de tarea, de rol o de producto de trabajo.[GUIA]

Propiedades de los Elementos de Desglose de Trabajo

En EPFC las propiedades (columnas) comunes para todos los tipos de WBE (Elementos de Desglose de Trabajo) son las siguientes:

- Nombre de presentación (Presentation name).
- Índice (Index): número secuencial de 0 en adelante que sirve para identificar cada elemento (WBE).
- Predecesores (Predecessors): lista de valores index de los WBE que son predecesores inmediatos.
- Información del modelo (Model info): indica si el WBE está basado en un elemento de contenido y el tipo de herencia. En algunos casos indica el subtipo de elemento de contenido, por ejemplo, entrada obligatoria (mandatory input) para un producto de trabajo.
- Tipo (Type): los tipos de WBE son proceso para despliegue (delivery process), patrón de proceso (capability pattern), actividad (activity), iteración (iteration), fase (phase), hito (Milestone) y descriptor de tarea (task descriptor).
- Ocurrencias múltiples (Multiple occurrences): sí/no permitir más de una ocurrencia (instancia) del proceso.
- Opcional (Optional): si está desactivado significa que se puede quitar dicho WBE del proceso sin problemas.
- Planificado (Planned): sí/no incluirlo para exportar a una herramienta de gestión de proyectos.
- Suprimido (Suppresed): sí/no queda excluido de la publicación del proceso.

Los WBE de tipo actividad o descriptor de tarea tienen estas otras propiedades adicionales:

- Condicionado por sucesos ("Event-driven"): activado si la tarea/actividad se inicia cuando ocurre cierto evento; desactivado si se inicia después de otra(s) tareas(s) (relación de precedencia entre ellas).
- Continuo ("Ongoing"): activado si la tarea/actividad es continua a lo largo de toda la vida de la instancia del proceso; desactivado si tiene un inicio y final en el proceso.
- Repetible ("Repeatable"): activado si la tarea/actividad puede iterar para una instancia del proceso; desactivado si puede ocurrir una sola vez cuando el proceso es llevado a cabo.

Las anteriores propiedades se pueden seleccionar en las columnas del mismo nombre en las perspectivas de desglose, o también en la pestaña "General" de la vista "Propiedades" para el elemento seleccionado.

En la pestaña "Propiedades" (Properties) de un WBS es posible incluir información más completa y precisa para los WBE del WBS, que completan la heredada desde los elementos reutilizados.

En la pestaña "General" (General), la propiedad Dependencia (Dependency) permite precisar el tipo de las relaciones de precedencia inmediata entre los elementos de un WBS, eligiendo entre:

- Finalizar para Iniciar (Finish To Start), que es la opción por defecto;
- Finalizar para Finalizar (Finish To Finish);
- Iniciar para Finalizar (Start To Finish); o
- Iniciar para Iniciar (Start To Start).

Establecer los tipos de dependencias es especialmente útil a la hora de exportar el proceso a una herramienta de gestión de proyectos:

En la pestaña "Documentación" (Documentation) se pueden asignar valores a las propiedades: prefijo ("prefix"), descripción breve ("brief description"), guía de utilización ("usage guidance"), factores clave ("key considerations"), y descripción perfeccionada ("refined description").

En la pestaña Roles ("Roles") la propiedad asistido por ("assisted by") sirve para indicar otros roles que apoyan al realizador principal ("primary performer").

En la pestaña Productos de trabajo ("Work products") la propiedad entradas externas ("external inputs") permite añadir entradas externas al proceso, es decir, que no se generan como salida de ninguna actividad del proceso.

Además del WBS, EPFC permite otras tres vistas o perspectivas de los elementos de desglose de un elemento de proceso. Una manera diferente de visualizar la misma información: los elementos y asociaciones incluidos en la estructura de desglose de un cierto elemento de proceso.

En la figura se ve un ejemplo del uso de la perspectiva de “Asignación de Equipos” (“team allocation”) para ver o editar la asignación de roles a cada actividad y los productos de trabajo que cada rol tiene asociados.

A continuación se ve un ejemplo de vista de “Utilización de Productos de Trabajo”, ésta define como se pueden establecer los productos de trabajo que serán creados y utilizados en cada actividad del proceso, e identificar los roles responsables de dichos productos de trabajo. Además, para cada producto de trabajo se incluyen las propiedades, en forma de columnas, de: estado de entrada (“entry state”), estado de salida (“exit state”) y entregable (“deliverable”), también llamado producto final.

En la siguiente figura la Vista Consolidada (“consolidated view”) permite ver (pero no editar) de forma conjunta la lista de roles y productos de trabajo asociados con cada actividad. [GUIA]

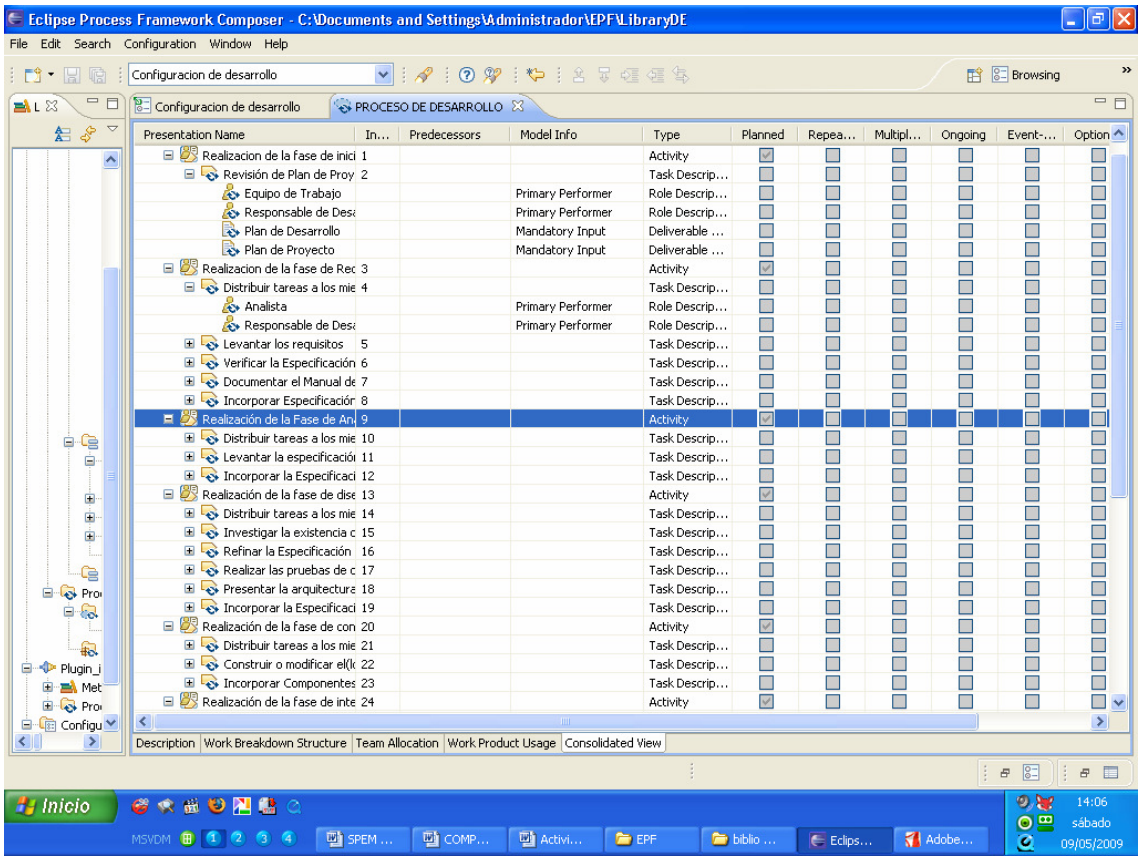

Ilustración 27. Vista consolidada en EPF.

3.2.6 Reutilización y adaptación de Actividades

Una forma de crear procesos es abordar la especificación de un proceso de abajo hacia arriba. Primero se definen patrones de proceso más sencillos (no incluyendo jerarquía de actividades, sino simplemente descriptores de tarea, de rol y de producto de trabajo). Después, se crean patrones más complejos en los que se reutilizan los patrones más sencillos definidos anteriormente. De esta forma se van componiendo los procesos reutilizando bloques de proceso cada vez más complejos hasta llegar a construir el proceso para despliegue.

EPFC ofrece dos opciones para reutilizar actividades (normalmente del tipo patrón de proceso) previamente creadas:

a) Arrastrar la actividad desde la ventana “Configuración” al elemento padre de la estructura de desglose donde se desee aplicar.

b) Hacer click derecho en el elemento padre y seleccionar “Aplicar patrón”.

EPFC incluye opciones diferentes para reutilizar actividades, aunque ofrecen al ingeniero de procesos la misma funcionalidad:

• Copiar (“Copy”): Se copia el contenido de la actividad. La copia se desconecta del original y se puede modificar para personalizarlo según exija el contexto del proceso en el que se aplica. Las modificaciones en la copia no afectan a la actividad original. Las modificaciones en la actividad original no se reflejan en las copias que se hayan hecho de dicha actividad en otros procesos.

• Extender (“Extend”): Se hereda el contenido de la actividad. Se pueden añadir nuevos elementos a la actividad heredada, pero no se puede modificar el contenido heredado. Los cambios realizados en la actividad extendida no afectan a la actividad original. En cambio, los cambios realizados en la actividad original se reflejan en las actividades que la extienden.

• Copia en Profundidad: (“Deep Copy”): Con la opción anterior se copia una actividad tal cual, es decir, si dentro de ella existe un elemento que fue aplicado utilizando la opción extender, en el sitio donde se copia se respetará la relación de extensión. Mediante la copia en profundidad lo que se consigue es que todo el contenido de la actividad se copie. Incluso los elementos que se aplicarán por extensión en la actividad, al realizar la copia en profundidad serán copiados y desconectados de la actividad original.

Una vez aplicada (reutilizada) una actividad se pueden modificar sus propiedades. Por ejemplo, es posible cambiar su documentación, añadir instrucciones o cambiar el tipo de la actividad para que sea una fase, iteración o actividad. Todo esto se hace en la vista “Propiedades” mediante las pestañas “General”, “Documentación e Instrucciones”.

Al aplicar una actividad puede haber elementos que no interesen en el contexto en que se va a utilizar dicha actividad. Para ello, en actividades copiadas (primera opción de las tres anteriores) se pueden eliminar los elementos que no interesen. En cambio, en actividades extendidas no es posible hacerlo. Pero si se quiere que un elemento no aparezca publicado en un proceso completo se puede hacer mediante la propiedad “Suprimir” que aparece al hacer click derecho sobre el elemento. El elemento suprimido no se eliminará del patrón, pero no se publicará.

El tipo de variabilidad (amplía, contribuye o reemplaza) de un elemento de proceso de tipo actividad (fase, iteración, actividad, patrón de capacidad) se especifica en la pestaña “General” de la vista “Propiedades”.

Para adecuar una metodología a proyectos específicos se aprovecha la capacidad de reutilización y la distinción entre elementos de método y elementos de método en uso (descriptores de elemento) ofrecida por SPEM y EPFC. Mediante dicha distinción, cada aparición de un elemento de método en un proceso se convierte en un descriptor de ese elemento, el cual se puede modificar para adaptarlo al contexto específico en el que aparece (es decir, las características del proceso o proyecto específico).

Es importante respetar lo máximo posible las capacidades ofrecidas por SPEM 2. Por ello, ya que SPEM ofrece los descriptores de elementos para poder adaptar un mismo elemento de método a varios contextos (en este caso el de cada proyecto específico), en lugar de crear nuevos elementos de método, con el contenido específico del contexto del proyecto en el que se aplican, es mejor utilizar dicho mecanismo de herencia y copia para adaptar los elementos (tareas, productos, etc.) definidos para el proceso general, a un tipo específico de proyecto. Un caso diferente es que el proyecto específico requiera elementos que no están contemplados en la metodología general. En este caso sí que es inevitable tener que crear dichos elementos.

Una vez se ha definido la estructura del proceso, se pueden definir los diagramas de actividad, de detalle de actividad y de dependencias de productos de trabajo para cada patrón de proceso

Asociados a un WBS, o a un WBE dentro de un WBS, con EPFC se pueden generar tres de los tipos de diagramas previstos en SPEM 2. Dichos diagramas los genera automáticamente EPFC, que además incluye un editor gráfico para mejorarlos y completarlos. El editor de la tabla de la estructura de desglose y el editor de diagramas son bidireccionales, en el sentido de que los cambios realizados con uno se trasladan automáticamente a la información mostrada por el otro.

Para abrir o crear un diagrama con EPFC, se selecciona un elemento de tipo actividad, se hace click derecho, se elige la opción "**Diagramas**" y se selecciona el tipo de diagrama que se quiere crear. Mediante la opción "Diagramas->Opciones de publicación*"* se pueden seleccionar los diagramas que se desean publicar.

También se pueden utilizar diagramas diferentes de los tres predefinidos con la herramienta.

Para ello, los diagramas realizados con otras herramientas se deben incluir en un tipo de guía llamado "Material de soporte" y seleccionar dicha guía como diagrama mediante la opción

"Diagramas"->Diagramas definidos por el usuario".

Una vez se han creado los patrones de proceso necesarios se pueden crear los procesos para despliegue reutilizando dichos patrones de proceso. La reutilización de patrones de proceso en procesos para despliegue se hace de igual forma que la reutilización de patrones de proceso dentro de otros patrones comentada en el punto anterior. También se pueden definir diagramas para el proceso para despliegue.

Se puede implementar una jerarquía tarea-actividad subproceso-proceso de la siguiente forma: Se crearían los patrones básicos de las actividades que sólo incluyen tareas. Después se crearían los patrones de los subprocesos en los que se reutilizan los patrones de las actividades. Finalmente se crearían los procesos como procesos para despliegue en los que se reutilizan los patrones de los subprocesos.

Una vez que se tienen todos los procesos y todos los elementos de método definidos se pueden publicar. [EPF03]

3.2.7 Publicación de contenidos

EPF Composer permite publicar los procesos y todo su contenido en un sitio Web para que éste pueda ser accedido con facilidad.

Para publicar los contenidos en EPF Composer se deben crear antes una o más configuraciones, ya que es con base a dichas configuraciones que la herramienta genera el

esquema de presentación y la organización del contenido en el sitio Web. Una vez que se tienen las configuraciones, se indica con base en cual se quiere publicar el contenido y luego la herramienta genera automáticamente todos los archivos necesarios para tener un sitio Web listo para montar en algún servidor Web. [EPF03]

Las opciones ofrecidas en cuanto a preferencias de publicación son las siguientes:

- Vía de acceso predeterminada: Seleccionar el directorio donde se guardará la Web generada.
- URL de información de retorno: Seleccionar la Web o dirección e-mail de retorno de información. En todas las páginas publicadas se incluye una dirección cuyo propósito es que se puedan realizar comentarios, informar de errores, etc., sobre el contenido publicado.
- Incluir el contenido del método en las páginas del descriptor: Si se marca esta opción, las descripciones realizadas en el contenido de método también se incluirán en las páginas de los descriptores. Si no se marca, cada página de descriptor incluirá un enlace a la página del elemento de método al que referencia para poder consultar sus objetivos, descripciones, etc.
- Diagramas de roles*:* En este apartado se puede ajustar el espaciado horizontal y vertical entre los elementos mostrados en los diagramas de roles y el número de líneas que puede tener el texto de esos elementos. Estos parámetros se deben ajustar en función de la longitud de los nombres de las tareas y los productos con los que están relacionados los roles.
- Diagramas de actividad*:* En este apartado se pueden configurar dos opciones relacionadas con la publicación de diagramas de actividad:
    - Publicar diagramas de actividad para extensiones de actividad que no se hayan modificado: Si no se marca esta opción, las páginas de las actividades que extiendan otra actividad y no se hayan modificado, no incluirán el diagrama de actividad. La página cuenta con un enlace a la página de la actividad que extienden, donde se puede consultar el diagrama. Si se marca la casilla, el diagrama de actividad también aparecerá en la página de la actividad que extiende.
    - Publicar diagramas de detalle de actividad que no se hayan creado en el editor de procesos: Si se marca esta opción también se publicarán los diagramas por defecto.

Las opciones que se pueden elegir sobre la publicación son las siguientes:

- Seleccionar la configuración que se va a publicar.
- Seleccionar contenido de la configuración: publicar todos los procesos o sólo los que se seleccionen.
- Seleccionar opciones de publicación.

3.2.8 SPEM y EPF COMPOSER

Una herramienta basada en SPEM sería una herramienta para crear y personalizar procesos como es el caso de EPF Composer en donde la mayoría de los elementos presentes en ella se pueden ver definidos en el metamodelo SPEM.

Cuando se va a definir y a documentar un proceso en EPF Composer, lo más recomendable es definir o concebir todos los contenidos que se incluirán en la composición del proceso de acuerdo a los lineamientos presentes en la documentación de la herramienta sobre como definir un proceso. En este caso estará muy relacionado con el metamodelo SPEM, en el que se basa la herramienta.

La ingeniería de métodos es un enfoque por el cual las metodologías se diseñan y construyen para cada proyecto concreto, aprovechando y reutilizando componentes metodológicos que se encuentran recopilados en una base de datos o repositorio. El contenido de este repositorio está soportado por la experiencia de los usuarios y de la organización. La aplicación de una metodología determinada requiere un metamodelo subyacente, es decir un conjunto de conceptos básicos (más las relaciones entre ellos) que determinan los bloques conceptuales básicos con los que se va a trabajar. En el caso de EPF, el metamodelo de procesos subyacente es SPEM2.0 (Software Process Engineering Metamodel). Una vez definida toda una metodología, y cuando así se considere necesario, el usuario puede generar una documentación que englobe a aquellos procesos que considere determinantes para el funcionamiento de su organización o para la puesta en marcha de un proyecto concreto. La documentación es impresa, con formato HTML y bajo un esquema determinado, en el sitio Web que el usuario haya establecido y está compuesta por un número de páginas que definen tanto a los procesos como a los elementos que los componen.

Un metamodelo tiene importante uso en sistemas o entornos abiertos o heterogéneos, donde se trata de crear capas de mayor nivel de abstracción que tengan meta información necesaria para manejar situaciones más abstractas o generales. [EPF03]

3.3 COMPETISOFT: Mejora de Procesos para PYMES

Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica

Las PyMES encuentran dificultades al querer aplicar un programa de mejoras de Procesos de Software, ya que deben hacer grandes inversiones de dinero, tiempo y recursos. Además hay modelos complejos de aplicar y los beneficios no se ven a corto plazo.

COMPETISOFT, es un modelo aplicable a pequeñas y medianas empresas y sirve para usar de modelo de referencia para mejorar los procesos y la calidad de los productos, no tiene altos costos, y es fácil de entender y aplicar. Pretende ser la base para alcanzar evaluaciones exitosas con otros modelos o normas, tales como ISO 9000:2000 o CMM V1.1.

COMPETISOFT v 0.2 presenta el "Modelo de Referencia de Procesos", el "Modelo de Evaluación" y el "Modelo de Mejora de Procesos" de COMPETISOFT para PyMES iberoamericanas, dedicadas al desarrollo de software, para fomentar la mejora de prácticas en la gestión de Ingeniería del Software y así alcanzar mejores niveles internacionales de competitividad. [COMPE02]

El objetivo principal del proyecto COMPETISOFT es incrementar el nivel de competitividad de las PyMES Iberoamericanas productoras de software mediante la creación y difusión de un marco metodológico común que, ajustado a sus necesidades específicas, pueda llegar a ser la base sobre la que establecer un mecanismo de evaluación y certificación de la industria del software reconocido en toda Iberoamérica.

Para lograr ese objetivo general se plantean los siguientes objetivos específicos:

- Desarrollar un Marco Metodológico común ajustado a la realidad socio-económica de las PYMES iberoamericanas, orientado a la mejora continua de sus procesos. Este Marco Metodológico, que estará compuesto por un Modelo de Procesos, un Modelo de Capacidades y un Método de Evaluación, será validado, en el marco de este proyecto mediante su aplicación controlada, en empresas y organizaciones de diferentes países de la región CYTED.

- Difundir la cultura de la mejora de procesos en el sector informático iberoamericano y más específicamente formar, tanto a investigadores y/o docentes universitarios (formación de formadores) como a profesionales de un buen número de PyMES productoras de software, mediante los cursos que se organizaran en este proyecto CYTED y mediante la difusión, a través de la Web del proyecto, de los materiales de formación que se elaborarán; así como mediante la supervisión y desarrollo de tesis de postgrado para estudiantes y docentes de la región.

- Incidir en los diferentes organismos de normalización y certificación de los países iberoamericanos, para que asuman que los principios metodológicos objeto de este proyecto CYTED pueden ser la base para establecer un mecanismo común y mutuamente reconocido de evaluación y certificación de la industria del software Iberoamericana.

El modelo de procesos de COMPETISOFT está basado en el definido por MoProSoft. Tiene tres categorías de procesos: Alta Dirección, Gerencia y Operación que reflejan la estructura de una organización.

3.3.1 Patrón de procesos

El patrón de procesos es un esquema de elementos que servirá para la documentación de los procesos. Está constituido por tres partes: "Definición general del proceso", "Prácticas y Guías de ajuste".

En la "Definición general del proceso" se identifica su nombre, categoría a la que pertenece, propósito, descripción general de sus actividades, objetivos, indicadores, metas cuantitativas, responsabilidad y autoridad, subprocesos en caso de tenerlos, procesos relacionados, entradas, salidas y productos internos.

En las "Prácticas" se identifican los roles involucrados en el proceso y las competencias requeridas, se describen las actividades en detalle, asociándolas a los objetivos del proceso, se presenta un diagrama de flujo de trabajo, se describen las verificaciones y validaciones requeridas, se listan los productos que se incorporan a la base de conocimiento, se identifican los recursos de infraestructura necesarios para apoyar las actividades, se establecen las mediciones del proceso.

En las "Guías de ajuste" se sugieren modificaciones al proceso que no deben afectar los objetivos del mismo.

El patrón de procesos fue utilizado como esquema para documentar los procesos de COMPETISOFT.

El patrón de procesos que utilicen las organizaciones puede ser distinto del sugerido en este modelo, pero debe de preservar los objetivos, indicadores y metas cuantitativas correspondientes para lograr el propósito general de COMPETISOFT.

El patrón de proceso puede ser utilizado para documentar e integrar otros procesos que no fueron contemplados en el modelo. [COMPE01]

3.3.2 Definición general de procesos

Proceso	Nombre de proceso, precedido por el acrónimo establecido en la definición de los elementos de la estructura del modelo de procesos.
Categoría	Nombre de la categoría a la que pertenece el proceso y el acrónimo entre paréntesis.
Propósito	Objetivos generales medibles y resultados esperados de la implantación efectiva del proceso.
Descripción	Descripción general de las actividades y productos que componen el flujo de trabajo del proceso.
Objetivos	Objetivos específicos cuya finalidad es asegurar el cumplimiento del propósito del proceso. Los objetivos se identifican como O1, O2, etc.

Indicadores	Definición de los indicadores para evaluar la efectividad del cumplimiento de los objetivos del proceso. Los indicadores se identifican como I1, I2, etc. y entre paréntesis se especifica una o más identificaciones de los objetivos a los que dan respuesta.
Metas cuantitativas	Valor numérico o rango de satisfacción por indicador.
Responsabilidad y autoridad	Responsabilidad es el rol principal responsable por la ejecución del proceso. Autoridad es el rol responsable por validar la ejecución del proceso y el cumplimiento de su propósito.
Subprocesos (opcional)	Lista de procesos de los cuales se compone el proceso en cuestión.
Procesos relacionados	Nombres de los procesos relacionados.

Ilustración 28. Definición general del proceso. [COMPE01]

Entradas

Nombre	**Fuente**
Nombre del producto o recurso	Referencia al origen del producto o recurso

Ilustración 29. Entradas. [COMPE01]

Salidas

Nombre	**Descripción**	**Destino**	**Plantilla Soporte**	**Forma de aprobación**
Nombre del producto o recurso	Descripción y características del producto o recurso	Referencia al destinatario del producto o recurso	Documento que especifica los lineamientos que debe satisfacer la salida	Identificación de la verificación, validación o descripción de otra forma de aprobación, en caso de no requerirse alguna de éstas escribir la palabra Ninguna. Estas aprobaciones definen el momento a partir del cual el producto estará bajo control de Conocimiento de la Organización

Ilustración 30. Salida. [COMPE01]

Productos internos

Nombre	Descripción	Plantilla Soporte	Forma de aprobación
Nombre del producto generado y utilizado en el propio proceso	Descripción y características del producto	Documento que especifica los lineamientos que debe satisfacer la salida	Identificación de la verificación, validación o descripción de otra forma de aprobación, en caso de no requerirse alguna de éstas, escribir la palabra Ninguna. Estas aprobaciones definen el momento a partir del cual el producto estará bajo control de Conocimiento de la Organización

Ilustración 31. Productos internos. [COMPE01]

Prácticas

Roles involucrados y competencias.
Identificación de roles involucrados y competencias requeridas.

Abreviatura	Rol	Competencias
Abreviatura del rol	Nombre del rol	Competencias de tipo técnico y aptitudes requeridas por el rol para poder ejecutar el proceso de forma adecuada

Ilustración 32. Prácticas. [COMPE01]

Actividades

Se asocian a los objetivos y describen las tareas y roles responsables.

Rol	Descripción
A1. Nombre de la actividad (O1, O2, ...)	
Entradas	
Abrev. del(de los) rol(es)	A1.1 Descripción de tarea 1. Si la actividad es una verificación o validación se hará referencia a la identificación de la misma.
	A1.2 Descripción de tarea 2
Salidas	
A2. Nombre de la actividad (O1, O2, ...)	
Entradas	

	A2.1 Descripción de tarea 1
	A2.2 Descripción de tarea 2
Salidas	

Ilustración 33. Actividades. [COMPE01]

Guías de ajuste
Descripción de posibles modificaciones al proceso que no deben afectar los objetivos del mismo.

Identificación de la guía	Descripción
Identificación de la guía	Descripción

Ilustración 34. Guías de ajuste. [COMPE01]

3.3.3 Niveles de capacidad de Procesos

En la siguiente tabla se pueden ver las correspondencias entre los niveles de capacidad de proceso y los colores en que se marcan los productos de entrada, salida e internos y toda la sección de prácticas del modelo de procesos.

Nivel	Capacidad de proceso	Color
1	Realizado	amarillo
2	Gestionado	Azul
3	Establecido	Verde
4	Predecible	Rosa
5	Optimizado	ninguno

Ilustración 35. Niveles de Procesos de COMPETISOFT. [COMPE01]

A continuación se describen las convenciones del marcaje que se adoptaron y su interpretación:

Los productos y las prácticas marcadas en amarillo son los que se recomiendan como primeros a implementar para poder cumplir con el nivel 1 de capacidades de procesos.

Una vez implementadas éstas, se sugiere agregar los productos y las prácticas marcadas en azul y así sucesivamente agregando lo marcado en verde y rosa. El nivel 5 se logra cuando de manera sistemática a través del tiempo se cumple los objetivos y se logra mejorar las metas cuantitativas de los procesos. Esta parte de la definición de los procesos quedó sin ninguna marca en particular.

Algunos productos o prácticas quedaron marcados de varios colores. Esto significa que algunas partes de los productos o de las prácticas se requieren en niveles distintos. La interpretación que se sugiere es que cuando se implementa un nivel más bajo de capacidad se toma en cuenta nada más las partes marcadas en el color correspondiente y cuando se sube de nivel se incorporan las partes del nivel siguiente.

Los productos que aparecen en las entradas marcados de un color también incluyen todos los elementos de este producto que corresponden a los niveles más bajos.

Los roles quedaron marcados en el color del nivel más bajo en el que tengan asignadas las actividades como su responsabilidad.

En el caso de la práctica "Incorporación a la Base de Conocimiento" algunos productos que se generan desde nivel 1 quedaron marcados con niveles más altos que corresponden al nivel a que se les aplican las verificaciones o validaciones. Sin embargo, estos productos deben de incorporarse a la "Base de Conocimiento" desde el nivel 1 cuando todavía no fueron verificados ni validados. [COMPE01]

3.3.4 Definición general del Proceso de Desarrollo de Software

Proceso	**OPE.2 Desarrollo de Software**
Categoría	Operación (OPE)
Propósito	El propósito de Desarrollo de Software es la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software nuevos cumpliendo con los requisitos especificados y con las normativas de seguridad de información.
Descripción	El proceso de Desarrollo de Software se compone de uno o más ciclos de desarrollo. Cada ciclo está compuesto de las siguientes fases, cada fase debe incorporar controles de seguridad de información: • Inicio: Revisión del *Plan de Desarrollo* por los miembros del *Equipo de Trabajo* para lograr un entendimiento común del proyecto y para obtener el compromiso de su realización. • Requisitos: Conjunto de actividades cuya finalidad es obtener la documentación de la *Especificación de Requisitos* y *Plan de Pruebas de Sistema,* para conseguir un entendimiento común entre el cliente y el equipo del proyecto. • Análisis y Diseño: Ambas fases involucran un conjunto de actividades en las cuales se analizan los requisitos especificados para producir una descripción de la estructura de los *Componentes*, la cual servirá de base para la construcción. Involucra la concepción de la arquitectura o diseño de alto nivel y la especificación detallada considerando los lineamientos y decisiones para incluir atributos de calidad del producto y seguridad de la solución. Como resultado se obtiene el Documento de *Especificación del Sistema* y el *Plan de Pruebas de Integración.* • Construcción: Conjunto de actividades para producir *Componente(s)* de software que correspondan con el *Análisis y Diseño,* así como la realización de pruebas unitarias. Como

	resultado se obtienen el (los) *Componente(s)* de software probados. • Integración Conjunto de actividades para integrar y probar los *Componentes* de software, basadas en el *Plan de Pruebas de Integración* , con la finalidad de obtener el *Sistema de Software* que satisfaga la *Especificación del Sistema* establecida. Se revisan los datos de entrada, el control de procesos internos, la integridad de los mensajes, la validación de los datos de salida y protección de los datos de prueba. Como resultado se obtiene el *Sistema de Software* para ser probado. • Pruebas. Conjunto de actividades para probar el *Sistema de Software*, basadas en el *Plan de Pruebas de Sistema*, con la finalidad de obtener el *Sistema de Software* que satisfaga los requisitos especificados. Se genera la versión final del *Manual de Usuario*, *Manual de Operación* y *Manual de Mantenimiento*. Como resultado se obtiene el *Sistema de Software* probado y documentado. • Cierre: Integración final de la *Configuración de Software* generada en las fases para su entrega. Identificación y documentación de las *Lecciones Aprendidas.* Generación del *Reporte de Mediciones y Sugerencias de Mejora.* Para generar los productos de cada una de estas fases se realizan las siguientes actividades: • Distribución de tareas, se asignan las responsabilidades a cada miembro del *Equipo de Trabajo* de acuerdo al *Plan de Desarrollo*. • Producción, verificación, validación o prueba de los productos, así como su corrección correspondiente. Se limita las modificaciones o cambios al sistema, se consideran los controles de seguridad sobre las mejoras solicitadas y se realizan a través del Procedimiento de Control de Cambios. • Generación del *Reporte de Actividades.*
Objetivos	O1 Lograr un entendimiento de las necesidades del cliente por el equipo de trabajo y estar de acuerdo con la solución propuesta mediante la ejecución de las actividades de la Fase de Requisitos. O2 Lograr que los productos de salida sean consistentes con los productos de entrada en cada fase de un ciclo de desarrollo mediante las actividades de verificación, validación o prueba, y mediante el control de cambios de los mismos. O3 Llevar a cabo las actividades de las fases de un ciclo mediante el cumplimiento del *Plan de Desarrollo* actual. O4 Garantizar que al final del desarrollo del sistema todos los requisitos funcionales estén trazados a los *Componentes*.

	O5 Sustentar la realización de ciclos posteriores o proyectos de mantenimiento futuros mediante la integración de la *Configuración de Software* del ciclo actual.
Indicadores	I1 (O1) El cliente ha entendido, participado y aprobado la solución propuesta por el equipo de trabajo. I2 (O2) En cada fase de un ciclo se efectúan todas las actividades de verificación, validación o prueba, así como las correcciones correspondientes. I3 (O3) Las actividades planificadas en cada fase de un ciclo se realizan conforme a lo establecido en el *Plan de Desarrollo*. I4 (O4) Todos los requisitos funcionales se pueden rastrear en la *Matriz de trazabilidad* a nivel de componente. I5 (O5) La *Configuración de Software* está integrada por los productos generados en el ciclo.
Metas cuantitativas	Valor numérico o rango de satisfacción por indicador: Ejemplos: M1 que el 100% de los requisitos definidos han sido contemplados en la especificación del *Sistema de Software*. M2 que el 100% de los criterios de calidad definidos hayan sido contemplados en la especificación del *Sistema de Software*. M3 que el 80% del grupo técnico haya revisado y aprobado el documento de *Especificación del Sistema*. M4 que el 100% de productos generados en cada fase hayan sido revisados y aprobados
Responsabilidad y autoridad	Responsable: • Responsable de Desarrollo de Software Autoridad: • Responsable de Administración del Proyecto Específico
Procesos relacionados	Administración de un Proyecto Específico Gestión de Conocimiento

Ilustración 36. Definición general de un Proceso de Software. [COMPE01]

Entradas

Nombre	Fuente
Plan de Proyecto • *Descripción del producto* • *Objetivos del Proyecto* • *Alcance* • *Entregables* • *Necesidad de negocio* • *Supuestos y premisas* • *Restricciones.*	Administración de un Proyecto Específico
Plan de Desarrollo • *Proceso Específico* • *Equipo de Trabajo* • *Calendario*	Administración de un Proyecto Específico

Ilustración 37. Entradas del Proceso de Desarrollo. [COMPE01]

Salidas

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
Especificación de Requisitos	Se compone de una introducción y una descripción de requisitos. Introducción: Descripción general del *Sistema de software* y su uso en el ámbito de negocio del cliente. Descripción de requisitos: * Funcionales: Necesidades establecidas que debe satisfacer el *Sistema de Software* cuando es usado en condiciones específicas. Las funcionalidades deben ser adecuadas, exactas y	Administración de un Proyecto Específico	No tiene plantilla	Ver1, Val1

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
	seguras. * Interfaz con usuario: Definición de aquellas características de la interfaz de usuario que permiten que el *Sistema de Software* sea fácil de entender, aprender, que genere satisfacción y con el cual el usuario pueda desempeñar su tarea eficientemente. Incluyendo la descripción del prototipo de la interfaz. * Interfaces externas: Definición de las interfaces con otro software o con hardware. * Confiabilidad: Especificación del nivel de desempeño del *Sistema de Software* con respecto a la madurez, tolerancia a fallas y recuperación. * Eficiencia: Especificación del nivel de desempeño del *Sistema de Software* con respecto al tiempo y a la utilización de recursos. * Mantenimiento: Descripción de los elementos que facilitarán la comprensión y la realización de las modificaciones futuras del *Sistema de Software*. * Portabilidad: Descripción de las características del *Sistema de Software* que permitan su transferencia de un ambiente a otro.			

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
	* Restricciones de diseño y construcción: Necesidades impuestas por el cliente. * Legales y reglamentarios: Necesidades impuestas por leyes, reglamentos, entre otros.			
Especificación del Sistema	Este documento contiene la descripción textual y gráfica de la estructura de los *Componentes* de software. El cual consta de las siguientes partes: Arquitectónica: Contiene la estructura interna del sistema, es decir la descomposición del sistema en subsistemas. Así como la identificación de los componentes que integran los subsistemas y las relaciones de interacción entre ellos. Incluye los controles de seguridad que se han diseñado que contendrá el *Sistema de Software* a desarrollar. Detallada: Contiene el detalle de los componentes que permita de manera evidente su construcción y prueba en el ambiente de programación.	Administración de un Proyecto Específico	No tiene Plantilla	Ver5, Val2, Ver7 Val3
Componente	Conjunto de unidades de código relacionadas.	Administración de un Proyecto Específico	No tiene plantilla	Prueba Unitaria Exitosa
Sistema de Software	Conjunto de componentes agrupados en subsistemas, posiblemente anidados.	Administración de un Proyecto Específico	No tiene plantilla	Prueba de integración exitosa, prueba de

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
				sistema exitosa
Configuración de Software	Conjunto consistente de productos de software, que incluye: • *Especificación de Requisitos* • *Especificación del Sistema Software* • *Prototipo de la Interfaz de Usuario* • *Matriz de Trazabilidad* • *Plan de Pruebas de Sistema* • *Reporte de Pruebas de Sistema* • *Reporte de Pruebas de Aceptación* • *Plan de Pruebas de Integración* • *Reporte de Pruebas de Integración* • *Plan de Pruebas de Seguridad* • *Reporte de Pruebas de Seguridad* • *Nivel de Seguridad alcanzado por el Sistema de Software.* • *Manual de Usuario* • *Manual de Operación* • *Manual de Mantenimiento* • *Sistema de Seguimiento de Defectos* • *Casos de Prueba del Sistema*	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Manual de Usuario	Documento electrónico o impreso que describe la forma de uso del *Sistema de*	Administración de un Proyecto Específico	No tiene plantilla	Ver4 Ver11

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
	Software con base a la interfaz del usuario. Éste deberá ser redactado en términos comprensibles a los usuarios. Incluirá los controles de seguridad que el usuario debe conocer y sobre lo cuales debe responsabilizarse.			
Manual de Operación	Documento electrónico o impreso que contenga la información indispensable para la instalación y administración del *Sistema de software*, así como el ambiente de operación (sistema operativo, base de datos, servidores, etc.), parametrización y configuración de seguridad. Éste deberá ser redactado en términos comprensibles al personal responsable de la operación.	Administración de un Proyecto Específico	No tiene plantilla	Ver11
Prototipo de Interfaz de Usuario	Primera aproximación a la interfaz de la herramienta que va a usar el usuario.	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Manual de Mantenimiento	Documento electrónico o impreso que describe la *Configuración de Software* y el ambiente usado para el desarrollo y pruebas (compiladores, herramientas de análisis y diseño, construcción y pruebas). Este deberá ser redactado en términos comprensibles al personal de mantenimiento.	Administración de un Proyecto Específico	No tiene plantilla	Ver13
Reporte de Actividades	Registro periódico de actividades, fechas de inicio y fin, responsables y mediciones, tales como: • tiempo de producción, de corrección, de	Administración de un Proyecto Específico	No tiene plantilla	Ninguna

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
	verificación y de validación, • defectos encontrados en verificación, validación o prueba, implementación de controles de seguridad, • tamaño de productos.			
Lecciones Aprendidas	Registro de mejores prácticas, problemas recurrentes y experiencias exitosas en la solución de problemas, encontrados en un ciclo de desarrollo.	Gestión de Conocimiento	No tiene plantilla	Ninguna
Reporte de Mediciones y Sugerencias de Mejora	Registro que contiene: • Mediciones de los indicadores del proceso de Desarrollo de Software (ver **Mediciones**). • Sugerencias de mejora al proceso de Desarrollo de Software (métodos, herramientas, formatos, estándares, etc.). • Recomendaciones para alcanzar los niveles de seguridad estándar.	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Matriz de Trazabilidad	Relación entre los requisitos, elementos análisis y diseño, componentes y planes de pruebas.	Administración de un Proyecto Específico	No tiene plantilla	Ver5 Ver9
Plan de Pruebas de Sistema	Identificación de pruebas requeridas para el cumplimiento de los requisitos especificados	Administración de un Proyecto Específico	No tiene plantilla	Ver2

Nombre	**Descripción**	**Destino**	**Plantilla Soporte**	**Forma de aprobación**
Reporte de Pruebas de Sistema	Registro de participantes, fecha, lugar, duración y de defectos encontrados.	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Reporte de Pruebas de Aceptación	Registro de participantes, fecha, lugar, duración y de defectos de aceptación encontrados.	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Plan de Pruebas de Integración	Descripción que contiene: * El orden de integración de los componentes o subsistemas, guiado por la parte arquitectónica del *Análisis y Diseño.* * Pruebas que se aplicarán para verificar la interacción entre los componentes.	Administración de un Proyecto Específico	No tiene plantilla	Ver5
Reporte de Pruebas de Integración	Registro de participantes, fecha, lugar, duración y de defectos encontrados.	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Sistema de Seguimiento de Defectos	Registro en el que se anotan todos los defectos encontrados en el producto.	Administración de un Proyecto Específico	No tiene Plantilla	Ninguna
Casos de Prueba del Sistema	Conjunto de todas las pruebas ejecutadas sobre el sistema. Incluye también las pruebas de Aceptación y las de Seguridad.	Administración de un Proyecto Específico	No tiene plantilla	Ver10
Plan de Pruebas de Seguridad	Descripción que contiene: *Controles implementados a probar *Parametrización a probar *Resultados esperados	Administración de un proyecto Específico	No tiene plantilla	Ver3
Reporte de Pruebas de Seguridad	Registro de participantes, fecha, lugar, duración y de defectos encontrados.	Administración de un Proyecto Específico	No tiene plantilla	Ninguna
Nivel de Seguridad Alcanzado	Registro de los controles de seguridad implementados a satisfacción y su brecha contra los requisitos de	Administración de un Proyecto Específico	No tiene Plantilla	Ninguna

Nombre	Descripción	Destino	Plantilla Soporte	Forma de aprobación
	seguridad solicitados como estandar			

Ilustración 38. Salidas del Proceso de Desarrollo. [COMPE01]

Productos internos

Nombre	Descripción	Plantilla Soporte	Forma de aprobación
Reporte(s) de Verificación	Registro de participantes, fecha, lugar, duración y defectos encontrados.	Reporte de Verificación	Ninguna
Reporte(s) de Validación	Registro de participantes, fecha, lugar, duración y defectos encontrados.	Reporte de Validación	Ninguna

Ilustración 39. Productos internos del Proceso de Desarrollo. [COMPE01]

Roles

Roles involucrados y competencias

Identificación de roles involucrados y competencias requeridas.

Abreviatura	Rol	Competencias
RAPE	Responsable de la Administración del Proyecto Específico	Capacidad de liderazgo con experiencia en la toma de decisiones, planificación estratégica, manejo de personal y desarrollo de software.
RD	Responsable de Desarrollo de Software	Conocimiento y experiencia en el desarrollo de software.
AN	Analista	Conocimiento y experiencia en la obtención, especificación y análisis de los requisitos.
DU	Diseñador de la Interfaz de Usuario	Conocimiento en diseño de interfaces de usuario y criterios ergonómicos.
DI	Diseñador	Conocimiento y experiencia en el diseño de la estructura de los componentes de software.
PR	Programador	Conocimiento y/o experiencia en la programación, integración y

		pruebas unitarias.
RPU	Responsable de Pruebas	Conocimiento y experiencia en la planificación y realización de pruebas de integración y de sistema.
RE	Revisor	Conocimiento en las técnicas de revisión y experiencia en el desarrollo de software.
RM	Responsable de Manuales	Conocimiento en las técnicas de redacción y experiencia en el desarrollo de software.
ET	Equipo de Trabajo	Conocimiento y experiencia de acuerdo a su rol.
CL	Cliente	Interpretación del estándar de la especificación de requisitos.
US	Usuario	Ninguna
AR	Arquitecto	Conocimiento de la plataforma tecnológica objetivo, conocimiento de los recursos existentes que pueden ser reutilizados, visión global del negocio y de las soluciones de arquitectura que garantizan la evolución del sistema
ST	Soporte Técnico	Conocimiento de la plataforma objetivo y de los lineamientos existentes en la empresa cliente para el despliegue de componentes y la operación de sistema
RS	Responsable de Seguridad	Responsable de establecer los requisitos de seguridad de información estándar y el nivel alcanzado por el software desarrollado.
ES	Equipo de Seguridad	Responsable de instalar, probar e identificar el nivel de seguridad alcanzado.

Ilustración 40. Roles del Proceso de Desarrollo. [COMPE01]

Actividades
Se asocian a los objetivos y describen las tareas y roles responsables.

Rol	Descripción
A1. Realización de la Fase de Inicio (O3)	
Entradas	*Plan de Desarrollo*
RD ET	A1.1. Revisar con los miembros del equipo de trabajo el *Plan de Proyecto* y el *Plan de Desarrollo* actual para lograr un entendimiento común y obtener su compromiso con el proyecto.
RD	A1.2. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.
Salidas	*Reporte de Actividades*
A2. Realización de la Fase de Requisitos (O1, O3)	
Entradas	*Plan de Desarrollo*
RD AN	A2.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
AN CL US ES	A2.2. Levantar los requisitos. • Identificar y consultar fuentes de información (clientes, usuarios, sistemas previos, documentos, etc.) para obtener nuevos requisitos. • Realizar reuniones de trabajo con los usuarios seleccionados para levantar los requisitos • Analizar los requisitos identificados para delimitar el alcance y su factibilidad, considerando las restricciones del ambiente del negocio del cliente o del proyecto. • Identificar y establecer los requisitos de calidad del producto requeridos de acuerdo a las condiciones existentes • Elaborar el prototipo de la interfaz con el usuario cuando sea necesario. • Hacer trazabilidad entre los requisitos y los objetivos del sistema • Negociar los requisitos con los involucrados • Generar o actualizar la *Especificación de Requisitos*.
RE DU	A2.3. Verificar la *Especificación de Requisitos* (Ver1)
AN DU	A2.4. Corregir los defectos encontrados en la *Especificación de Requisitos* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
CL US RPU	A2.5. Validar la *Especificación de Requisitos* (**Val1**).
AN DU	A2.6. Corregir los defectos encontrados en la *Especificación de Requisitos* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones.
RPU AN	A2.7. Elaborar o modificar *Plan de Pruebas de Sistema*.
RE	A2.8. Verificar el *Plan de Pruebas de Sistema* (**Ver2**).

RPU	A2.9. Corregir los defectos encontrados en el *Plan de Pruebas de Sistema* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
ES	A2.10. Elaborar o modificar el *Plan de Pruebas de Seguridad*.
RS	A2.11. Verificar el Plan de Pruebas de Seguridad (**Ver3**).
AN ES	A2.12. Corregir los defectos encontrados en el *Plan de Pruebas de Seguridad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RM	A2.13. Documentar la versión preliminar del *Manual de Usuario* o modificar el manual existente.
RE	A2.14. Verificar el *Manual de Usuario* (**Ver4**).
RM	A2.15. Corregir los defectos encontrados en el *Manual de Usuario* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RD	A2.16. Incorporar *Especificación de Requisitos*, *Plan de Pruebas de Sistema* y *Manual de Usuario* a la *Configuración de Software*.
RD	A2.17. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.
Salidas	*Especificación de Requisitos* *Plan de Pruebas del Sistema* *Plan de Pruebas de Seguridad* *Manual de Usuario* *Reporte de Actividades*
A3. Realización de la Fase de Análisis (O1, O2, O3, O4)	
Entradas	*Plan de Desarrollo*
RD AN DI	A3.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
AN DI DU AR ST	A3.2. Levantar la *Especificación del Sistema* • Elaborar el modelo conceptual que representa las entidades relevantes del sistema. • Analizar la *Especificación de Requisitos* para modelar las unidades funcionales del sistema. • Actualizar las matrices de trazabilidad de los requisitos con las unidades funcionales del sistema. • Especificar el detalle de la apariencia y el comportamiento de la interfaz con base en la *Especificación de Requisitos* de forma que se puedan prever los recursos para su implementación. • Especificar los niveles de calidad de servicio requeridos para cada unidad funcional. • Definir relevancia de implementación de las unidades funcionales según

	su impacto en la arquitectura. • Generar la *Especificación del Sistema*. • Generar la *Matriz de Trazabilidad*.
RE	A3.3. Verificar la *Especificación del Sistema* y la *Matriz de Trazabilidad* (**Ver5**).
AN DI DU	A3.4. Corregir los defectos encontrados en la *Especificación del Sistema* y en la *Matriz de Trazabilidad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
CL RPU	A3.5. Validar la *Especificación del Sistema* (**Val2**).
AN DI DU	A3.6. Corregir los defectos encontrados en la *Especificación del Sistema* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones.
RPU	A3.7. Elaborar o modificar *Plan de Pruebas de Integración*.
RE	A3.8. Verificar el *Plan de Pruebas de Integración* (**Ver6**).
RPU	A3.9. Corregir los defectos encontrados en el *Plan de Pruebas de Integración* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RD	A3.10. Incorporar la *Especificación del Sistema*, *Matriz de Trazabilidad* y *Plan de Pruebas de Integración* a la *Configuración de Software*.
RD	A3.11. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.
Salidas	*Especificación del Sistema* *Matriz de Trazabilidad* *Plan de Pruebas de Integración* *Configuración del Software* *Reporte de Actividades*
A4. Realización de la Fase de Diseño (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD AN DI	A4.1. Planificar las tareas de diseño de alto nivel y distribuirlas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
RAPE, RD	A4.2. Investigar la existencia de componentes para su reutilización en el proyecto
AN, Di, DU, AR ST	A4.3. Refinar la *Especificación del Sistema,* • Analizar la *Especificación del Sistema* para generar la descripción de la estructura interna del sistema y su descomposición en subsistemas, y estos a su vez en componentes, definiendo las interfaces entre ellos. • Definir las tácticas de arquitectura a utilizar para satisfacer niveles de calidad del servicio.

	• Identificar las soluciones alternativas y definir la arquitectura candidata del sistema. • Identificar los componentes reutilizables que serán aplicados al producto. • Definir la plataforma tecnológica en la que se implementará el sistema • Describir el detalle de los componentes que permita su construcción de manera evidente. • Realizar el modelo de datos para los objetos que requieren persistencia • Actualizar la *Especificación del Sistema* para que incluya las decisiones de arquitectura. • Actualizar la *Matriz de Trazabilidad.*
AR	A4.4. Realizar las pruebas de concepto de la arquitectura tecnológica para asegurar el cumplimiento de los atributos de calidad.
RD, CL	A4.5. Presentar la arquitectura candidata al cliente y lograr su aprobación
RE	A4.6. Verificar la *Especificación del Sistema* y la *Matriz de Trazabilidad* (**Ver7**).
AN DI DU	A4.7. Corregir los defectos encontrados en la *Especificación del Sistema* y en la *Matriz de Trazabilidad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
CL RPU	A4.8. Validar la *Especificación del Sistema* (**Val3**).
AN DI	A4.9. Corregir los defectos encontrados en la *Especificación del Sistema* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones.
RAPE RD	A4.10. Realizar la estimación de construcción de cada unidad funcional.
RD	A4.11. Incorporar la *Especificación del Sistema*, *Matriz de Trazabilidad* a la *Configuración de Software*.
RD	A4.12. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.
RD	A4.13. Ajustar el equipo de trabajo de acuerdo a las demandas de construcción
Salidas	*Especificación del Sistema* *Matriz de Trazabilidad* *Configuración del Software* *Reporte de Actividades*
A5. Realización de la Fase de Construcción (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD	A5.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
PR	A5.2. Construir o modificar el(los) *Componente*(s) de software:

	• Implementar o modificar *Componente*(s) con base a la parte detallada de la *Especificación del Sistema*. • Definir el esquema de base de datos en el ambiente de desarrollo
PR RD	A5.3 Realizar pruebas unitarias • Definir y aplicar pruebas unitarias para verificar que el funcionamiento de cada componente esté acorde con la parte detallada de la *Especificación del Sistema*. • Corregir los defectos encontrados hasta lograr pruebas unitarias exitosas (sin defectos). • Actualizar la *Matriz de Trazabilidad,* incorporando los componentes construidos o modificados
RE	A5.3. Verificar la *Matriz de Trazabilidad* (**Ver9**).
PR	A5.4. Corregir los defectos encontrados en la *Matriz de Trazabilidad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.
RD	A5.5. Incorporar *Componentes* y Registro de Rastreo a la Configuración de Software.
Salidas	*Componente(s)* *Matriz de Trazabilidad* *Configuración del Software*
A6. Realización de la Fase de Integración (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD	A6.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
PR RPU	A6.2. Realizar integración y pruebas. • Verificar que todas las unidades funcionales están listas para su integración • Crear el procedimiento de generación del programa distribuible de acuerdo a la plataforma objetivo • Integrar todas las unidades funcionales de acuerdo al procedimiento definido • Aplicar las pruebas siguiendo el *Plan de Pruebas de Integración*, documentando los resultados en un *Reporte de Pruebas de Integración*. • Reportar los defectos encontrados en el Sistema de Seguimiento de Defectos. • Corregir los defectos encontrados hasta lograr una prueba de integración exitosa (pruebas de regresión) • Actualizar la *Matriz de Trazabilidad.*
RD	A6.3. Incorporar *Software*, *Reporte de Pruebas de Integración, Matriz de Trazabilidad,* a la *Configuración de Software*.
RD	A6.4. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.
Salidas	*Software*

	Reporte de Pruebas de Integración *Sistema de Seguimiento de Defectos* *Matriz de Trazabilidad* *Configuración del Software* *Reporte de Actividades*
A7. Realización de la Fase de Pruebas (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD	A7.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
RPU, ES	A.7.2 Diseñar los *Casos de Prueba del Sistema,* en base al *Plan de Pruebas del Sistema* y el *Plan de Pruebas de Seguridad* • Diseñar los casos de prueba funcionales • Diseñar los casos de prueba no funcionales (eficiencia, usabilidad, portabilidad, etc.) • Diseñar los casos de prueba de aceptación del sistema
RPU, ES	A7.3 Verificar los casos de prueba del sistema **(Ver10).** Reportar los defectos encontrados
RPU, RS	A7.4 Corregir los defectos encontrados en los *Casos de Prueba del Sistema* con base en el *Reporte de Verificación.*
RPU CL, RS	A7.5 Validar los *Casos de Prueba del Sistema* (**Val4**).
RPU	A7.6 Corregir los defectos encontrados en los *Casos de Prueba del Sistema* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones.
RPU	A7.7 Realizar pruebas del sistema en el ambiente de pruebas **(Ver10).** Instalar el producto en el ambiente de pruebas Preparar los scripts y datos de prueba Realizar las pruebas del sistema basadas en los casos de prueba diseñados Reportar los defectos encontrados en el *Sistema de Seguimiento de Defectos*
RPU	A7.9 Realizar las *Pruebas del Seguridad*: • Ejecutar los *Casos de Prueba del Sistema* siguiendo el *Plan de Pruebas de Seguridad.* • Reportar los defectos encontrados en el *Sistema de Seguimiento de Defectos* Identificar y comunicar el *Nivel de Seguridad Alcanzado.*
PR	A7.8 Realizar las correcciones de acuerdo al *Sistema de Seguimiento de Defectos*
PR, RPU,	A7.8 Realizar las *Pruebas de Aceptación del Sistema*

CL	• Ejecutar los *Casos de Prueba del Sistema*, en el entorno definido por el cliente siguiendo el *Plan de Pruebas del Sistema*, documentando los resultados en un *Reporte de Pruebas de Aceptación.* • Reportar los defectos encontrados en el *Sistema de Seguimiento de Defectos*
PR	A7.10 Corregir los defectos encontrados Realizar las correcciones de acuerdo al *Sistema de Seguimiento de Defectos*
RD	A7.11 Verificar y Cerrar los defectos • Verificar que las correcciones se realizaron y que los defectos pueden ser cerrados (pruebas de regresión) • Actualizar la *Matriz de Trazabilidad* para cerrar los defectos
RM	A7.12 Documentar el Manual de Operación o modificar el manual existente.
RE	A7.13 Verificar el Manual de Operación (Ver11).
RM	A7.14 Corregir los defectos encontrados en el Manual de Operación con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
RM	A7.15 Documentar el *Manual de Usuario* o modificar el existente.
RE	A7.16 Verificar el Manual de Usuario (Ver12).
RM	A7.18 Corregir los defectos encontrados en el Manual de Usuario con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
RD	A7.19 Incorporar *Matriz de Trazabilidad, Manual de Operación* y *Manual de Usuario* a la *Configuración de Software*.
RD	A7.20 Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.
Salidas	*Casos De Prueba del Sistema* *Reporte de Pruebas del Sistema* *Configuración de Software* *Reporte de Pruebas de Aceptación del Sistema* *Reporte de Pruebas de Seguridad* *Manual de Operación* *Matriz de Trazabilidad* *Sistema de Seguimiento de Defectos* *Manual de Usuario* *Reporte de Actividades*
A8. Realización de la Fase de Cierre (O2, O5)	
Entradas	
RM	A8.1. Documentar el *Manual de Mantenimiento* o modificar el existente.
RE	A8.2. Verificar el *Manual de Mantenimiento* (**Ver13**).
RM	A8.3. Corregir los defectos encontrados en el *Manual de Mantenimiento* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones.

RD	A8.4. Incorporar *Manual de Mantenimiento* a la *Configuración de Software*.
RM	A8.5. Capacitar al cliente en su entorno de trabajo • Definir temas y agenda de la capacitación • Definir los recursos que requiere la capacitación • Realizar la capacitación • Controlar la asistencia a la capacitación • Solicitar evaluación de la capacitación
RD ET	A8.6. Identificar las *Lecciones Aprendidas* e integrarlas a la *Base de Conocimiento*. Como ejemplo, se pueden considerar mejores prácticas, experiencias exitosas de manejo de riesgos, problemas recurrentes, entre otras.
RD ET	A8.7. Generar el *Reporte de Mediciones y Sugerencias de Mejora*
RD	A8.8. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas
RD	A8.9. Integrar la *Configuración del Software* como línea base.
Salidas	*Manual de Mantenimiento* *Configuración del Software* *Lecciones Aprendidas* *Reporte de Mediciones y Sugerencias de Mejora* *Reporte de Actividades* *Nivel de Seguridad Alcanzado*

Verificaciones y validaciones	Se definen las verificaciones y validaciones asociadas a los productos generados en las actividades que se mencionan. En la verificación como en la validación se identifican los defectos que deben corregirse antes de continuar con las actividades posteriores. La validación de un producto puede ser interna (dentro de la organización) o externa (por el cliente) con la finalidad de obtener su autorización. Se recomienda que las validaciones se efectúen una vez que las verificaciones asociadas al producto sean realizadas.

Verificación o Validación	**Actividad**	**Producto**	**Rol**	**Lineamientos de Verificación o Validación**
Ver1	A2.3	*Especificación de Requisitos*	RE	Verificar la claridad de redacción de la *Especificación de Requisitos* y su consistencia con la *Descripción del Producto* y con el estándar de documentación requerido en el *Proceso*

				Específico. Adicionalmente revisar que los requisitos sean completos y no ambiguos o contradictorios. En caso de que se haya establecido la usuabilidad como requisito de calidad del producto, revisar criterios ergonómicos como: retroalimentación inmediata, acciones mínimas, control de usuario, flexibilidad, protección contra errores, consistencia, corrección de errores. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Val1	A2.8	*Especificación de Requisitos*	CL, US, RP U	Validar que la *Especificación de Requisitos* cumple con las necesidades y expectativas acordadas. En caso de que se haya establecido la usuabilidad como requisito de calidad del producto, incluir la prueba de de la interfaz de usuario. • Selección de usuarios para la prueba, Diseño del cuestionario de perfil de usuario • Planteamiento de la hipótesis de usabilidad (script de prueba e instrumento) • Monitorear la prueba • Registrar la prueba • Hacer el cuestionario de usabilidad Los defectos encontrados se documentan en un *Reporte de Validación*.
Ver2	A2.11	*Plan de Pruebas de Sistema*	RE	Verificar consistencia del *Plan de Pruebas de Sistema* con la *Especificación de Requisitos* y con el estándar de documentación requerido en el *Proceso Específico*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Ver3	A2.14	*Plan de Pruebas de Seguridad*	RS	Verificar que el *Plan de Pruebas de Seguridad* es consistente con la *Especificación de Requisitos* Los defectos encontrados se documentan en

				un *Reporte de Verificación.*
Ver4	A2.17	*Manual de Usuario*	RE	Verificar consistencia del *Manual de Usuario* con la *Especificación de Requisitos* y con el estándar de documentación requerido en el *Proceso Específico*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Ver5	A3.3	*Especificación del Sistema* *Matriz de Trazabilidad*	RE, ES	Verificar la claridad de la documentación de la *Especificación del Sistema*, su factibilidad y la consistencia con la *Especificación de Requisitos* y con el estándar de documentación requerido en el *Proceso Específico*. Verificar que la *Matriz de Trazabilidad* contenga las relaciones adecuadas entre los requisitos y los elementos de la *Especificación del Sistema*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Val2	A3.5	*Especificación del Sistema*	CL, RPU, ES	Validar que la *Especificación del Sistema* cumple con las necesidades y expectativas acordadas con el cliente. Los defectos encontrados se documentan en un *Reporte de Validación*.
Ver6	A3.8	*Plan de Pruebas de Integración*	RE	Verificar consistencia del *Plan de Pruebas de Integración* con la *Especificación del Sistema* y con el estándar de documentación requerido en el *Proceso Específico*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Ver7	A4.6	*Especificación del Sistema* *Matriz de Trazabilidad*	RE, ES	Verificar que en la arquitectura incluida en la *Especificación del Sistema están representadas todas las unidades funcionales del sistema.* Verificar que la *Matriz de Trazabilidad* contenga las relaciones adecuadas entre los requisitos y los elementos de la *Especificación del Sistema*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Val3	A4.8	*Especificación del Sistema*	RAPE,	Validar que la Arquitectura de la *Especificación del Sistema* cumple con

			RD, ES	las necesidades especificadas en la etapa de análisis. Los defectos encontrados se documentan en un *Reporte de Validación*.
Ver8	A4.12	*Plan de Desarrollo*	RE	Verificar que los ciclos definidos corresponden con las necesidades de la arquitectura. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Ver9	A5.3	*Matriz de Trazabilidad*	RE	Verificar que la *Matriz de Trazabilidad* contenga las relaciones adecuadas entre los elementos de la *Especificación del Sistema* y los componentes. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Ver10	A7.3	*Casos de Prueba del Sistema*	AN, PR, DI, RS	Verificar que los *Casos de Prueba del Sistema* se ajustan al *Plan de Pruebas del Sistema* y al *Plan de Pruebas de Seguridad*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Val4	A7.5	*Casos de Prueba del Sistema*	AN, PR, DI, RS	Validar los *Casos de Prueba del sistema* con la *Especificación de Requisitos* para asegurar que las pruebas abarcan toda la funcionalidad definida por esta. Los defectos encontrados se documentan en un *Reporte de Validación*.
Ver11	A7.10	*Manual de Operación*	*RE*	*Verificar la consistencia del Manual de Operación* con el *Software* y con el estándar de documentación requerido en el *Proceso Específico.* Los defectos encontrados se documentan en un *Reporte de Verificación.*
Ver12	A7.13	*Manual de Usuario*	RE	Verificar consistencia del *Manual de Usuario* con el sistema de *Software* y con el estándar de documentación requerido en el *Proceso Específico*. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Ver13	A8.2	*Manual de Mantenimiento*	RE	Verificar consistencia del *Manual de Mantenimiento* con la *Configuración de Software* y con el estándar de documentación requerido en el *Proceso*

				Específico. Los defectos encontrados se documentan en un *Reporte de Verificación*.
Mediciones	Mediciones que se establecen para evaluar los indicadores del proceso. Las mediciones se identifican como M1, M2, etc. y entre paréntesis se especifica la identificación del indicador que le corresponde.			
Medición	**Indicador**	**Objeto de medición**	**Rol**	**Mecanismo de medición**
M1	I1	*Reportes de Verificación, Reportes de Validación y Reportes de Pruebas*	RAPE	Revisar los *Reportes de Verificación*, *Reportes de Validación* y/o reportes de pruebas de cada fase para la confirmación de que se han realizado estas actividades y se han incorporado las correcciones.
M2	I2	*Configuración del Software*	RD	Revisar la *Configuración de Software* para comprobar que los productos que la integran son los mismos que se generaron en el ciclo.
M3	I3	*Plan de Desarrollo Actual*	RD	Comparar el *Plan de Desarrollo* actual para cada fase con el *Reporte de Actividades* correspondiente para conocer la desviación contra *lo planificado*.
M4	I4	*Especificación del Sistema*	RD	Se comprueba que en la *Especificación del Sistema* se han tenido en cuenta todos los requisitos planteados.

Ilustración 41. Actividades del Proceso de Desarrollo. [COMPE01]

Guías de ajuste
Descripción de posibles modificaciones al proceso que no deben afectar los objetivos del mismo.

Requisitos: *Especificación de Requisitos*	La *Especificación de Requisitos* puede incluir un prototipo de interfaz con el usuario, sencillo, que inclusive no tenga funcionalidad del documento de requisitos de seguridad estándar.
Requisitos: *Manual de Usuario*	En la fase de Requisitos se puede omitir la elaboración o actualización del *Manual del Usuario*, así como su verificación. Sin embargo esta actividad se deberá realizar a más tardar en la fase de integración y pruebas.
Requisitos: *Plan de Pruebas de Sistema*	El *Plan de Pruebas de Sistema* se puede validar con el cliente, en caso que se acuerde con él y con el Equipo de Seguridad.

Análisis: ***Especificación del Sistema***	En caso que se acuerde con el cliente, se puede omitir la validación de la *Especificación del Sistema*, pero se debe validar con el Equipo de Seguridad.
Diseño: ***Especificación del Sistema***	En caso que se acuerde con el cliente, se puede omitir la validación del *Especificación del Sistema*.
Construcción: Revisión entre colegas del código	Antes de realizar pruebas unitarias se pueden incluir revisiones entre colegas para verificar el código de los componentes con respecto a la *Especificación del Sistema*. El beneficio de estas revisiones es la disminución del número de defectos de fases posteriores y el tiempo de corrección.
Construcción: Pruebas unitarias	Las pruebas unitarias se pueden definir de manera sistemática y documentada siguiendo el estándar IEEE Std 1008-1987 (R 1993) *Standard for Software Unit Testing.*
Construcción: Prototipo de interfaz	En la fase de Construcción se puede agregar la elaboración o modificación del prototipo de la interfaz para realizar una prueba con el usuario, con el fin de identificar defectos críticos de uso. Si no se cuenta con los usuarios para la prueba de interfaz puede recurrirse a la revisión de un experto o se pueden escoger individuos de un perfil similar.
Reporte de Actividades	Las mediciones requeridas en el *Reporte de Actividades* pueden ser modificadas de acuerdo a las necesidades de la organización o del proyecto.
Cierre	Cuando el APE y RD son la misma persona se omiten las actividades A8.7 y A.8.8 y se realizan conjuntamente con las actividades de cierre establecidas en APE.

Ilustración 42. Guías de ajuste del Proceso de Desarrollo. [COMPE01]

4. Desarrollo del PLUGIN

La idea de desarrollar este Plugin, surge con la motivación de obtener un módulo de Software, que sirva de complemento para sistemas más grandes, aumentando su funcionalidad, o aportando más datos que lo enriquezcan. De ésta forma se ayuda a las organizaciones a realizar sus actividades de manera más ordenada, organizada y eficiente.

Permite un mejor y más rápido acceso a la información y un desarrollo de las funciones organizativas y desarrolladoras, más eficaz y productivo.

A continuación se explicará detalladamente, los pasos que se siguieron para conseguirlo.

Como primer paso se estudió el Proceso de Desarrollo de Software definido en COMPETISOFT, y se definieron las entradas y salidas que corresponden a las tareas definidas en cada actividad del proceso. Se destacó de este modo las dependencias que surgen a partir de estas nuevas definiciones que dejan ver las entradas necesarias de cada actividad, producto de la salida de otras realizadas con anterioridad. En el modelo de COMPETISOFT están definidas en forma general, como entradas y salidas de cada actividad, y no en forma particular para cada una de las tareas que componen cada una de estas actividades.

Las entradas y salidas que se definieron están de color marrón, en el caso que una actividad esté conformada por tareas correspondientes a diferentes niveles de capacidad, se utilizó el color correspondiente al nivel de la tarea para pintar la entrada o salida de la misma.

Rol	**Descripción**
A1. Realización de la Fase de Inicio (O3)	
Entradas	*Plan de Desarrollo*
RD ET	Entrada: Plan de proyecto y Plan de desarrollo. A1.1. Revisar con los miembros del equipo de trabajo el *Plan de Proyecto* y el *Plan de Desarrollo* actual para lograr un entendimiento común y obtener su compromiso con el proyecto.
RD	A1.2. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas. Salida: Reporte de actividades
Salidas	*Reporte de Actividades*
A2. Realización de la Fase de Requisitos (O1, O3)	
Entradas	*Plan de Desarrollo*
RD AN	Entrada: Plan de desarrollo A2.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.

AN CL US ES	A2.2. Levantar los requisitos. • Identificar y consultar fuentes de información (clientes, usuarios, sistemas previos, documentos, etc.) para obtener nuevos requisitos. • Realizar reuniones de trabajo con los usuarios seleccionados para levantar los requisitos • Analizar los requisitos identificados para delimitar el alcance y su

	factibilidad, considerando las restricciones del ambiente del negocio del cliente o del proyecto. • Identificar y establecer los requisitos de calidad del producto requeridos de acuerdo a las condiciones existentes. • Elaborar el prototipo de la interfaz con el usuario cuando sea necesario. • Hacer trazabilidad entre los requisitos y los objetivos del sistema • Negociar los requisitos con los involucrados • Generar o actualizar la *Especificación de Requisitos*. Salida: Especificación de Requisitos
ui	Entradas: Especificación de Requisitos A2.3. Verificar la *Especificación de Requisitos* Salida: Especificación de requerimientos Reporte de verificación
AN DU	Entrada: Especificación de Requisitos y Reporte de verificación. A2.4. Corregir los defectos encontrados en la *Especificación de Requisitos* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Especificación de Requisitos
CL US RPU	Entrada: Especificación de Requisitos A2.5. Validar la *Especificación de Requisitos*. Salida: Especificación de Requisitos Reporte de validación
AN DU	Entrada: Especificación de Requisitos Reporte de validación A2.6. Corregir los defectos encontrados en la *Especificación de Requisitos* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones. Salida: Especificación de Requisitos
RPU AN	Entrada: Plan de Pruebas del sistema A2.7. Elaborar o modificar *Plan de Pruebas de Sistema*. Salida: Plan de Pruebas del sistema
RE	Entrada: Plan de Pruebas del sistema A2.8. Verificar el *Plan de Pruebas de Sistema*. Salida: Plan de Pruebas del Sistema Reporte de verificación
RPU	Entrada: Plan de Pruebas del sistema.

	Reporte de verificación A2.9. Corregir los defectos encontrados en el *Plan de Pruebas de Sistema* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Plan de Pruebas del sistema
ES	Entrada: Plan de Pruebas de Seguridad A2.10. Elaborar o modificar el *Plan de Pruebas de Seguridad*. Salida: Plan de Pruebas de Seguridad
RS	Entrada: Plan de Pruebas de Seguridad A2.11. Verificar el Plan de Pruebas de Seguridad. Salida: Plan de Pruebas de Seguridad Reporte de verificación
AN ES	Entrada: Plan de Pruebas de Seguridad Reporte de Verificación. A2.12. Corregir los defectos encontrados en el *Plan de Pruebas de Seguridad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Plan de Pruebas de Seguridad
RM	Entrada: Manual de Usuario A2.13. Documentar la versión preliminar del *Manual de Usuario* o modificar el manual existente. Salidas: Manual de usuario
RE	Entrada: Manual de Usuario A2.14. Verificar el *Manual de Usuario*. Salida: Manual de Usuario Reporte de verificación
RM	Entrada: Manual de Usuario Reporte de verificación. A2.15. Corregir los defectos encontrados en el *Manual de Usuario* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Manual de usuario
RD	Entrada: Especificación de Requisitos Manual de Usuario Configuración de Software Plan de Pruebas del Sistema

	A2.16. Incorporar *Especificación de Requisitos*, *Plan de Pruebas de Sistema* y *Manual de Usuario* a la *Configuración de Software*. Salida: Configuración de Software
RD	A2.17. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas. Salida: Reporte de actividades
Salidas	*Especificación de Requisitos* *Plan de Pruebas del Sistema* *Plan de Pruebas de Seguridad* *Manual de Usuario* *Reporte de Actividades*
A3. Realización de la Fase de Análisis (O1, O2, O3, O4)	
Entradas	*Plan de Desarrollo*
RD AN DI	Entrada: Plan de desarrollo A3.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
AN DI DU AR ST	A3.2. Levantar la *Especificación del Sistema* Entrada: Especificación de requisitos • Elaborar el modelo conceptual que representa las entidades relevantes del sistema. • Analizar la *Especificación de Requisitos* para modelar las unidades funcionales del sistema. • Actualizar las matrices de trazabilidad de los requisitos con las unidades funcionales del sistema. • Especificar el detalle de la apariencia y el comportamiento de la interfaz con base en la *Especificación de Requisitos* de forma que se puedan prever los recursos para su implementación. • Especificar los niveles de calidad de servicio requeridos para cada unidad funcional. • Definir relevancia de implementación de las unidades funcionales según su impacto en la arquitectura. • Generar la *Especificación del Sistema*, • Generar la *Matriz de Trazabilidad*. *Salida: Matriz de Trazabilidad*. Especificación del Sistema.
RE	Entrada: Especificación del Sistema Matriz de trazabilidad A3.3. Verificar la *Especificación del Sistema* y la *Matriz de Trazabilidad*. Salida: Reporte de verificación

AN DI DU	Entrada: Especificación del sistema , Matriz de Tarzabilidad, Reporte de verificación. A3.4. Corregir los defectos encontrados en la *Especificación del Sistema* y en la *Matriz de Trazabilidad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Especificación del Sistema y Matriz de Trazabilidad.
CL RPU	Entrada: Especificación del sistema A3.5. Validar la *Especificación del Sistema*. Salida: Reporte de verificación *Especificación del Sistema*
AN DI DU	Entrada: Especificación del Sistema Reporte de Validación A3.6. Corregir los defectos encontrados en la *Especificación del Sistema* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones. Salida: Especificación del Sistema
RPU	Entrada: Plan de Pruebas de Integración A3.7. Elaborar o modificar *Plan de Pruebas de Integración*. Salida: Plan de Pruebas de Integración
RE	Entrada: Plan de Pruebas de Integración A3.8. Verificar el *Plan de Pruebas de Integración*. Salida: Reporte de Verificación *Plan de Pruebas de Integración*
RPU	Entrada: Plan de Pruebas de Integración Reporte de Verificación. A3.9. Corregir los defectos encontrados en el *Plan de Pruebas de Integración* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Plan de Pruebas de Integración
RD	Entrada: Especificación del Sistema. Configuración de software Matriz de Trazabilidad Plan de Pruebas de Integración A3.10. Incorporar la *Especificación del Sistema*, *Matriz de Trazabilidad* y

	Plan de Pruebas de Integración a la *Configuración de Software*. Salida: Configuración del software.
RD	A3.11. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas. Salida: Reporte de actividades
Salidas	*Especificación del Sistema* *Matriz de Trazabilidad* *Plan de Pruebas de Integración* *Configuración del Software* *Reporte de Actividades*
A4. Realización de la Fase de Diseño (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD AN DI	Entrada: Plan de desarrollo. A4.1.Planificar las tareas de diseño de alto nivel y distribuirlas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
RAPE, RD	A4.2.Investigar la existencia de componentes para su reutilización en el proyecto

AN, Di, DU, AR ST	A4.3.Refinar la *Especificación del Sistema,* Entrada: Especificación del Sistema. • Analizar la *Especificación del Sistema* para generar la descripción de la estructura interna del sistema y su descomposición en subsistemas, y estos a su vez en componentes, definiendo las interfaces entre ellos. • Definir las tácticas de arquitectura a utilizar para satisfacer niveles de calidad del servicio. • Identificar las soluciones alternativas y definir la arquitectura candidata del sistema. • Identificar los componentes reutilizables que serán aplicados al producto. • Definir la plataforma tecnológica en la que se implementará el sistema • Describir el detalle de los componentes que permita su construcción de manera evidente. • Realizar el modelo de datos para los objetos que requieren persistencia. Actualizar la *Especificación del Sistema* para que incluya las decisiones de arquitectura. • Actualizar la *Matriz de Trazabilidad.* *Salida: Matriz de Trazabilidad* Especificación del sistema
AR	A4.4.Realizar las pruebas de concepto de la arquitectura tecnológica para asegurar el cumplimiento de los atributos de calidad.
RD, CL	A4.5.Presentar la arquitectura candidata al cliente y lograr su aprobación
RE	Entrada: Especificación del Sistema Matriz de Trazabilidad A4.6. Verificar la *Especificación del Sistema* y la *Matriz de Trazabilidad* Salida: Reporte de Verificación *Especificación del Sistema* *Matriz de Trazabilidad*
AN DI DU	Entrada: Especificación del Sistema, Matriz de Trazabilidad, Reporte de Verificación A4.7. Corregir los defectos encontrados en la *Especificación del Sistema* y en la *Matriz de Trazabilidad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Especificación del Sistema Matriz de Trazabilidad
CL RPU	Entrada: Especificación del sistema

	A4.8. Validar la *Especificación del Sistema*. Salida : Reporte de Validación *Especificación del Sistema*
AN DI	Entrada: Especificación del sistema Reporte de Validación A4.9. Corregir los defectos encontrados en la *Especificación del Sistema* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones. Salida: Especificación del Sistema
RAPE RD	A4.10. Realizar la estimación de construcción de cada unidad funcional.
RD	Entrada: Especificación del Sistema Matriz de Trazabilidad. Configuración de Software A4.11. Incorporar la *Especificación del Sistema*, *Matriz de Trazabilidad* a la *Configuración de Software*. Salida: Configuración de Software.
RD	A4.12. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas. Salida: Reporte de Actividades
RD	A4.13. Ajustar el equipo de trabajo de acuerdo a las demandas de construcción
Salidas	*Especificación del Sistema* *Matriz de Trazabilidad* *Configuración del Software* *Reporte de Actividades*
A5. Realización de la Fase de Construcción (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD	Entrada: Plan de Desarrollo. A5.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
PR	A5.2 .Construir o modificar el(los) *Componente*(s) de software: Entrada: Especificación del Sistema. Componentes • Implementar o modificar *Componente*(s) con base a la parte detallada de la *Especificación del Sistema*.

	• Definir el esquema de base de datos en el ambiente de desarrollo Salida: Especificación del sistema Componente
PR RD	Entradas: Especificación del Sistema Matriz de Trazabilidad Componentes A5.3. Realizar pruebas unitarias • Definir y aplicar pruebas unitarias para verificar que el funcionamiento de cada componente esté acorde con la parte detallada de la *Especificación del Sistema*. • Corregir los defectos encontrados hasta lograr pruebas unitarias exitosas (sin defectos). • Actualizar la *Matriz de Trazabilidad,* incorporando los componentes construidos o modificados Salida: Matriz de Trazabilidad
RE	Entrada: Matriz de Trazabilidad A5.4.Verificar la Matriz de Trazabilidad. Salida: Reporte de Verificación *Matriz de Trazabilidad*
PR	Entrada: Matriz de Trazabilidad Reporte de Verificación A5.5. Corregir los defectos encontrados en la *Matriz de Trazabilidad* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. Salida: Matriz de trazabilidad.

RD	Entrada: Componentes Configuración de Software A5.6. Incorporar *Componentes* y Registro de Rastreo a la Configuración de Software.

	Salida: Configuración de Software
Salidas	*Componente(s)* *Matriz de Trazabilidad* *Configuración del Software*
A6. Realización de la Fase de Integración (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD	Entrada: Plan de desarrollo. A6.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
PR RPU	A6.2. Realizar integración y pruebas. Entrada: Plan de Pruebas de Integración Sistema de Seguimiento de Defectos Matriz de Trazabilidad • Verificar que todas las unidades funcionales están listas para su integración. • Crear el procedimiento de generación del programa distribuible de acuerdo a la plataforma objetivo • Integrar todas las unidades funcionales de acuerdo al procedimiento definido • Aplicar las pruebas siguiendo el *Plan de Pruebas de Integración*, documentando los resultados en un *Reporte de Pruebas de Integración*. • Reportar los defectos encontrados en el Sistema de Seguimiento de Defectos. • Corregir los defectos encontrados hasta lograr una prueba de integración exitosa (pruebas de regresión) • Actualizar la *Matriz de Trazabilidad.* *Salida: Matriz de Trazabilidad actualizada* *Reporte de Pruebas de Integración* *Reporte de Prueba del sistema.*
RD	Entrada: Configuración de Software Sistema de Software Reporte de Pruebas de Integración Matriz de Trazabilidad A6.3 Incorporar *Software*, *Reporte de Pruebas de Integración, Matriz de Trazabilidad,* a la *Configuración de Software*. Salida: Configuración de Software
RD	A6.4 Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas. Salida: Reporte de actividades.

Salidas	*Software* *Reporte de Pruebas de Integración* *Sistema de Seguimiento de Defectos* *Matriz de Trazabilidad* *Configuración del Software* *Reporte de Actividades*
A7. Realización de la Fase de Pruebas (O1, O2, O3)	
Entradas	*Plan de Desarrollo*
RD	Entrada: Plan de desarrollo. A7.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al *Plan de Desarrollo* actual.
RPU, ES	A.7.2 Diseñar los *Casos de Prueba del Sistema*, en base a al *Plan de Pruebas del Sistema* y el *Plan de Pruebas de Seguridad* Entrada : Plan de Pruebas del Sistema Plan de Pruebas de Seguridad • Diseñar los casos de prueba funcionales • Diseñar los casos de prueba no funcionales (eficiencia, usabilidad, portabilidad, etc.) • Diseñar los casos de prueba de aceptación del sistema Salida: Casos de prueba del sistema
RPU, ES	A7.3 Verificar los casos de prueba del sistema. Entrada: Casos de Prueba del sistema Reportar los defectos encontrados Salida : Casos de Prueba del Sistema
RPU, RS	Entrada: Casos de Prueba del Sistema Reporte de Verificación A7.4 Corregir los defectos encontrados en los *Casos de Prueba del Sistema* con base en el *Reporte de Verificación.* *Salida:* Casos de Prueba del Sistema
RPU CL, RS	Entrada: Casos de Prueba del Sistema A7.5 Validar los *Casos de Prueba del Sistema.* Salida: Casos de Pruebas del Sistema Reporte de Validación
RPU	Entrada: Casos de Pruebas del Sistema Reporte de Validación A7.6 Corregir los defectos encontrados en los *Casos de Prueba del Sistema* con base en el *Reporte de Validación* y obtener la aprobación de las correcciones. *Salida:* Casos de Prueba del Sistema

RPU	A7.7 Realizar pruebas del sistema en el ambiente de pruebas. Entrada : Casos de Prueba del sistema Sistema de Seguimiento de Defectos Instalar el producto en el ambiente de pruebas Preparar los scripts y datos de prueba Realizar las pruebas del sistema basadas en los casos de prueba diseñados Reportar los defectos encontrados en el *Sistema de Seguimiento de Defectos* Salida: Sistema de Seguimiento de Defectos
RPU	A7.8 Realizar las *Pruebas de Seguridad*: Entradas: Plan de pruebas de Seguridad. Casos de prueba del sistema Sistema de Seguimiento de Defectos Ejecutar los *Casos de Prueba del Sistema* siguiendo el *Plan de Pruebas de Seguridad.* • Reportar los defectos encontrados en el *Sistema de Seguimiento de Defectos* Identificar y comunicar el *Nivel de Seguridad Alcanzado.* Salidas: Sistema de Seguimiento de Defectos *Nivel de Seguridad alcanzado*
PR	Entrada: Sistema de seguimiento de defectos. A7.9 Realizar las correcciones de acuerdo al *Sistema de Seguimiento de Defectos*
PR, RPU, CL	A7.10 Realizar las *Pruebas de Aceptación del Sistema* Entrada: *Casos de Prueba del Sistema,* *Plan de Pruebas del Sistema.* Sistema de seguimiento de defectos. • Ejecutar los *Casos de Prueba del Sistema*, en el entorno definido por el cliente siguiendo el *Plan de Pruebas del Sistema*, documentando los resultados en un *Reporte de Pruebas de Aceptación.* • Reportar los defectos encontrados en el *Sistema de Seguimiento de Defectos* *Salida: Sistema de seguimiento de defectos* *Reporte de Pruebas de Aceptación*
PR	A7.11 Corregir los defectos encontrados *Entrada: Sistema de seguimiento de defectos* Realizar las correcciones de acuerdo al *Sistema de Seguimiento de Defectos*

RD	A7.12 Verificar y Cerrar los defectos Entrada: Matriz de trazabilidad. • Verificar que las correcciones se realizaron y que los defectos pueden ser cerrados (pruebas de regresión) • Actualizar la *Matriz de Trazabilidad* para cerrar los defectos Salida: Matriz de trazabilidad actualizada. Reporte de Verificación
RM	Entrada: Manual de operación. A7.13 Documentar el Manual de Operación o modificar el manual existente. Salida: Manual de Operación
RE	Entrada: Manual de Operación A7.14 Verificar el Manual de Operación. Salida: Manual de Operación verificado Reporte de Verificación
RM	Entrada: Manual de Operación Reporte de Verificación A7.15 Corregir los defectos encontrados en el Manual de Operación con base en el Reporte de Verificación y obtener la aprobación de las correcciones. *Salida:* Manual de Operación
RM	Entrada: Manual de Usuario. A7.16 Documentar el *Manual de Usuario* o modificar el existente. Salida: Manual de Usuario
RE	Entrada: Manual de Usuario A7.17 Verificar el Manual de Usuario. Salida: Manual de Usuario Reporte de Verificación.
RM	Entrada: Manual de Usuario Reporte de Verificación A7.18 Corregir los defectos encontrados en el Manual de Usuario con base en el Reporte de Verificación y obtener la aprobación de las correcciones. *Salida:* Manual de Usuario
RD	Entrada: *Manual de Operación* *Manual de Usuario.* *Configuración de Software* *Matriz de Trazabilidad* A7.19 Incorporar, *Matriz de Trazabilidad, Manual de Operación* y *Manual de*

	Usuario a la *Configuración de Software*. Salida: Configuración de Software
RD	A7.20 Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas. Salida: Reporte de Actividades.
Salidas	*Casos De Prueba del Sistema* *Reporte de Pruebas del Sistema* *Configuración de Software* *Reporte de Pruebas de Aceptación del Sistema* *Reporte de Pruebas de Seguridad* *Manual de Operación* *Matriz de Trazabilidad* *Sistema de Seguimiento de Defectos* *Manual de Usuario* *Reporte de Actividades*
A8. Realización de la Fase de Cierre (O2, O5)	
Entradas	
RM	Entrada: Manual de Mantenimiento A8.1. Documentar el *Manual de Mantenimiento* o modificar el existente. Salida: Manual de Mantenimiento
RE	Entrada: Manual de Mantenimiento A8.2. Verificar el *Manual de Mantenimiento*. Salida: Manual de Mantenimiento Reporte de Verificación
RM	Entrada: Manual de Mantenimiento Reporte de Verificación A8.3. Corregir los defectos encontrados en el *Manual de Mantenimiento* con base en el *Reporte de Verificación* y obtener la aprobación de las correcciones. *Salida:* Manual de Mantenimiento
RD	Entrada: Manual de mantenimiento Configuración de Software A8.4. Incorporar *Manual de Mantenimiento* a la *Configuración de Software*. Salida: Configuración de Software
RM	A8.5. Capacitar al cliente en su entorno de trabajo • Definir temas y agenda de la capacitación • Definir los recursos que requiere la capacitación • Realizar la capacitación • Controlar la asistencia a la capacitación

	• Solicitar evaluación de la capacitación
RD ET	A8.6. Identificar las *Lecciones Aprendidas* e integrarlas a la *Base de Conocimiento*. Como ejemplo, se pueden considerar mejores prácticas, experiencias exitosas de manejo de riesgos, problemas recurrentes, entre otras. Salida :Lecciones aprendidas
RD ET	A8.7. Generar el *Reporte de Mediciones y Sugerencias de Mejora* *Salida: Reporte de Mediciones y Sugerencias de Mejoras*
RD	A8.8. Elaborar el *Reporte de Actividades* registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas Salida: Reporte de Actividades
RD	Entrada : *Configuración del Software* A8.9. Integrar la *Configuración del Software* como línea base. Salida : *Configuración del Software*
Salidas	*Manual de Mantenimiento* *Configuración del Software* *Lecciones Aprendidas* *Reporte de Mediciones y Sugerencias de Mejora* *Reporte de Actividades* *Nivel de Seguridad Alcanzado*

Ilustración 43. Definición del Proceso en el Plugin.

En primer lugar, es importante mencionar, que el procedimiento que se llevó a cabo para desarrollar este Plugin se basó en los pasos descriptos en la sección EPFC, que explica el uso de la herramienta y brinda algunas recomendaciones a tener en cuenta.

Inicialmente se creo una "Biblioteca de Métodos", llamada "LibraryDE", que contiene la definición del Plugin en forma completa, es decir, allí se encuentran las definiciones de los "Procesos" que se instanciaron y las "Configuraciones" realizadas.

Se construyó el Plugin, llamado "Plugin de Desarrollo", en base a las actividades que componen el Proceso de Desarrollo de COMPETISOFT y a la declaración de las entradas y salidas de cada una de ellas, que se muestran en el cuadro anterior.

Este Plugin contiene un "Method Content", con la definición de todos los "Elementos de Método", y la definición de los "Procesos" instanciados.

Para una mejor organización de la información se definió un "Paquete de Contenidos", llamado "Paquete de Desarrollo", donde se realizó la definición de todo los "Contenidos de Métodos", como Roles, Tareas y Entradas y Salidas. Respetando el patrón básico de SPEM: alguien (Rol) hace algo (Tarea) para obtener algo (Producto de Trabajo).

También se instanciaron las "Categorías" a las que corresponden cada uno de los elementos definidos con anterioridad.

De esta forma la información se encuentra mejor estructurada y organizada.

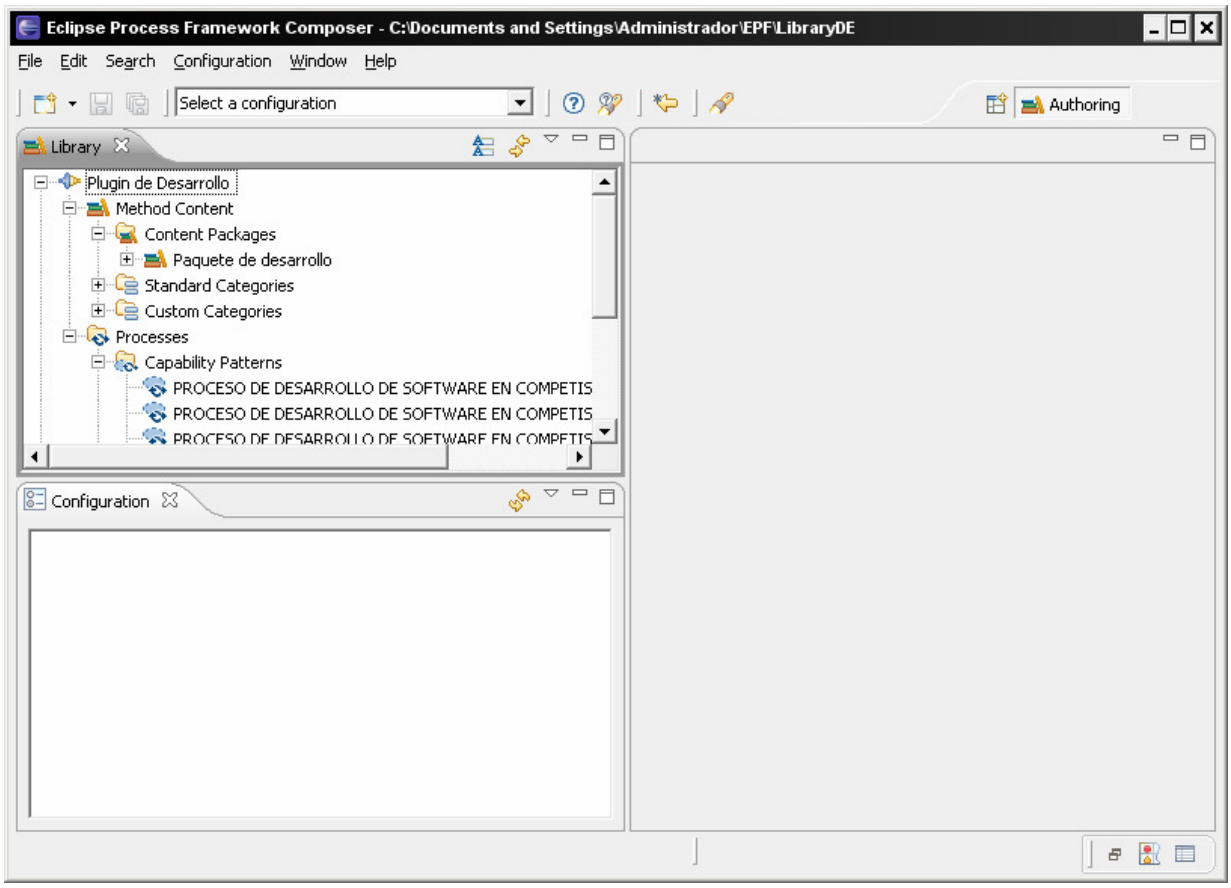

Ilustración 44. Plugin de Desarrollo.

Los primeros “Elementos de Método” que se definieron fueron los “Productos de Trabajo”, asociándoles a los mismos, la información principal que los describe (“Name”, “Presentation name” y “Brief description”). En los casos que se consideró necesario, como por ejemplo en el “Producto de Trabajo” definido como la “Especificación de Requisitos” se completo el campo Main Description, para ofrecer mayor información sobre el elemento instanciado.
Además se clasificaron según los tipos predefinidos por SPEM, estos son: “Artefacto”, “Entregable” y “Resultado”.
Los “Productos de Trabajo” llamados “Casos de prueba del Sistema”, “Componente”, “Configuración de Software” y “Sistema de Software” se clasificaron como “Entregables”, ya que su definición empaqueta otros “Productos de Trabajo” con fines de entrega a un cliente interno o externo. El resto se clasificó como “Artefactos”, por ser elementos de naturaleza tangible, como SPEM los define.
Estos “Productos de Trabajo” son los que se definieron en el cuadro de la ilustración 48 teniendo en cuenta las entredas y salidas que consumen y producen las actividades del Proceso de Desarrollo definido en COMPETISOFT.

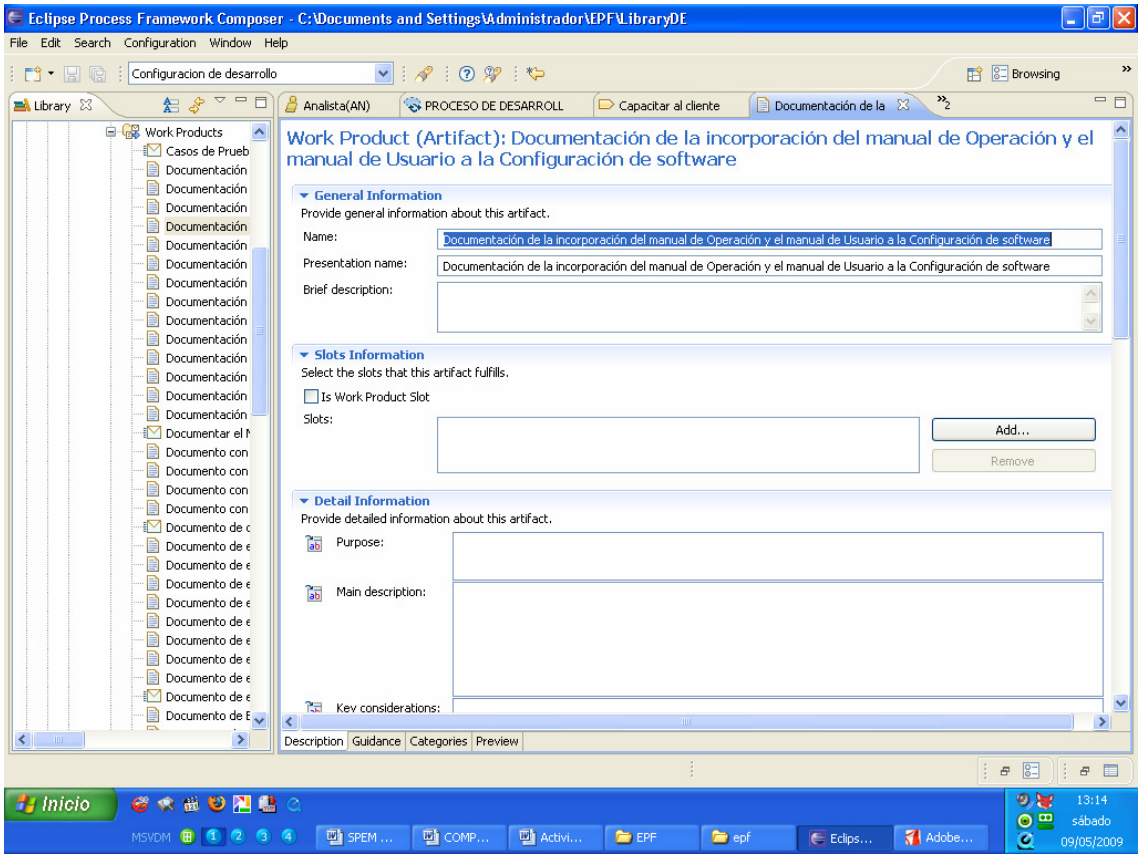

Ilustración 45. Producto de Trabajo en EPF.

A continuación, se definieron los "Roles". Se asoció a los mismos la siguiente información: "Name", "Presentation Name" y "Brief Description", además se les asoció los "Productos de Trabajo" con los que se relacionan. Estos roles y sus descripciones son los que asigna el Proceso de Desarrollo de COMPETISOFT para llevar a cabo las actividades.

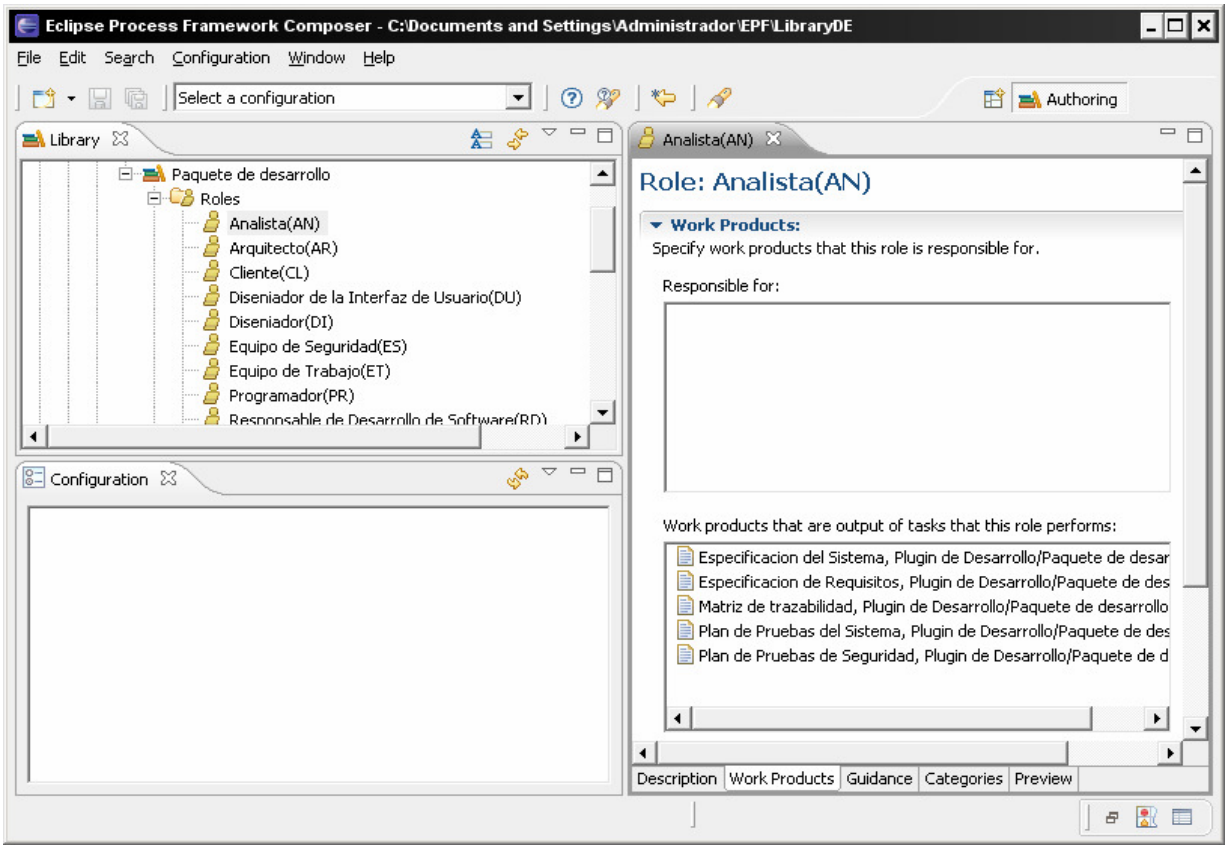

Ilustración 46. Roles y Productos de Trabajo asociados.

El siguiente paso fue la definición de las Tareas, se realizó una equivalencia con las actividades definidas en COMPETISOFT correspondientes a cada fase definida allí. A cada Tarea se le asoció los "Productos de Trabajo" que consume y produce, definiendo cuáles de ellos son obligatorios u opcionales. También se les asoció los Roles que las llevan a cabo, así como también los "Pasos" que las componen, según el Modelo de Procesos de COMPETISOFT.
De las mismas se informó el "Name", "Presentation Name" y "Brief Description".

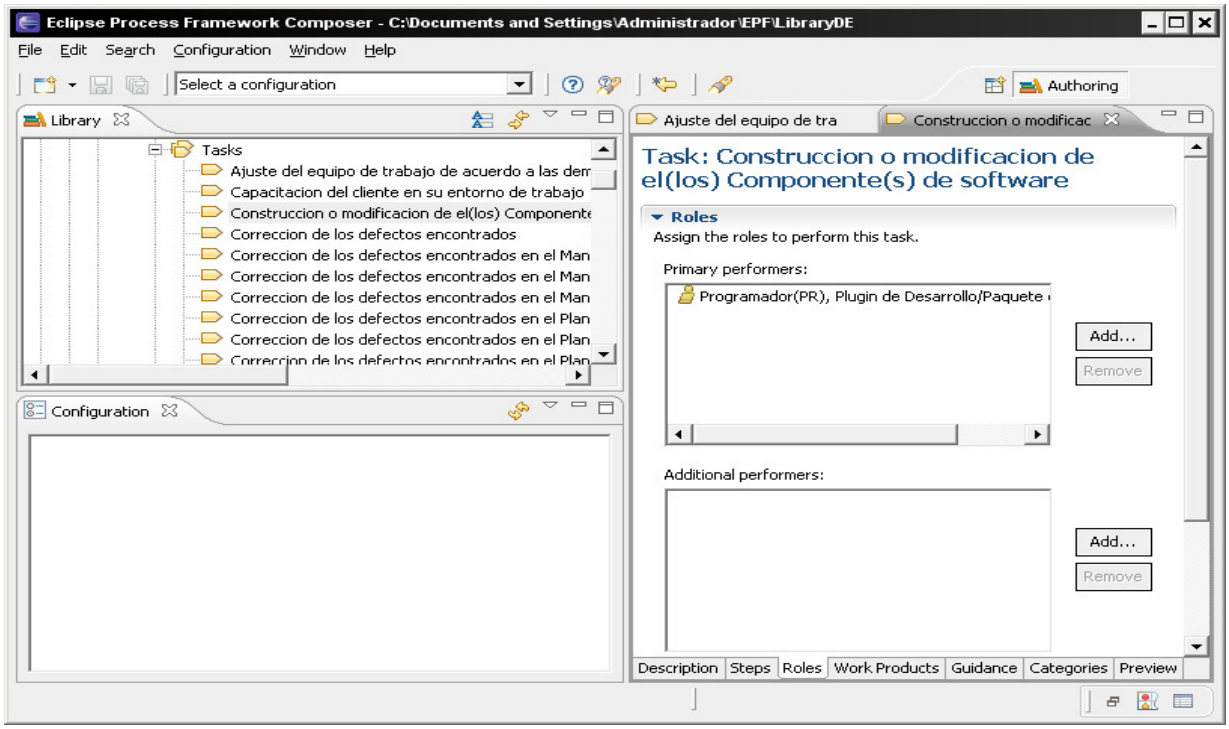

Ilustración 47. Roles asociados a las tareas.

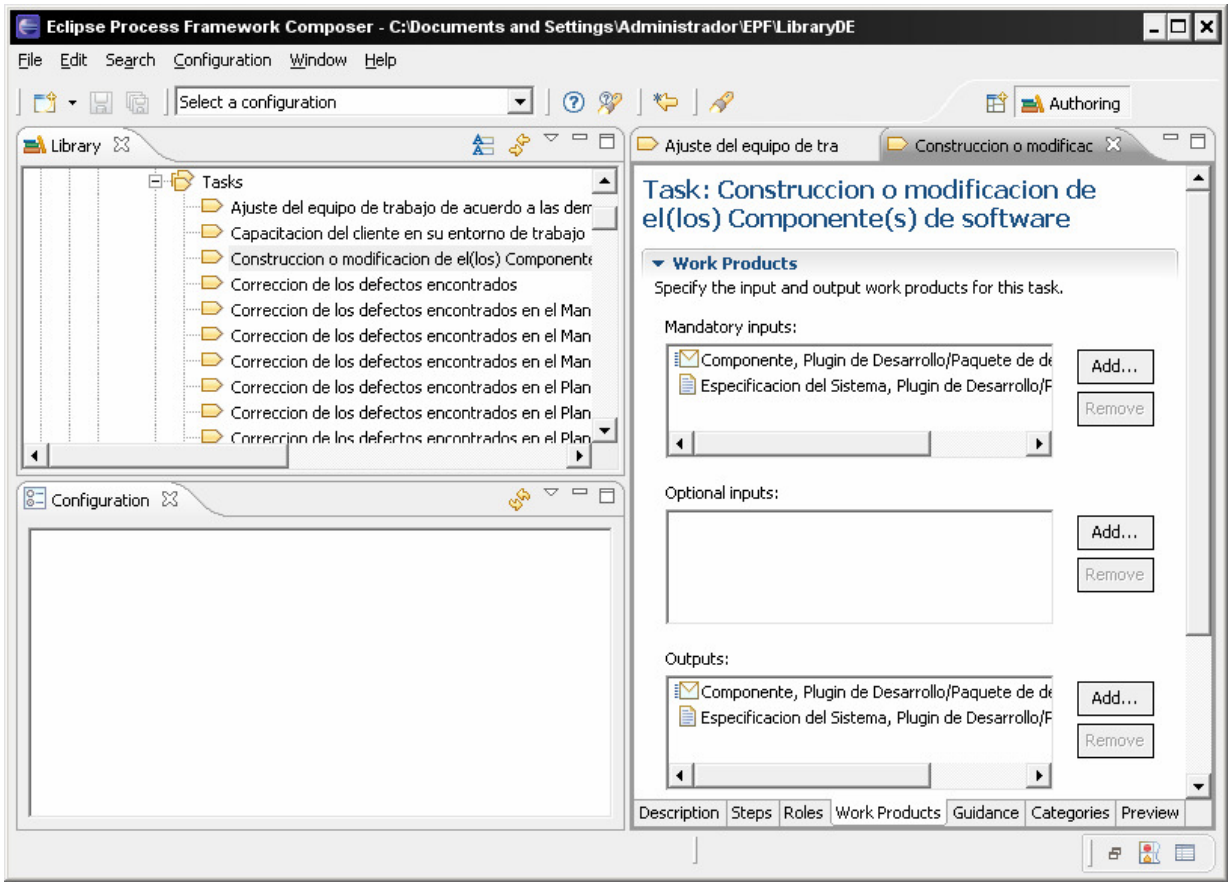

Ilustración 48. Productos de Trabajo asociados a las Tareas.

Luego de crear los “Elementos de Método”, se declararon las “Categorias” de elementos que permitieron clasificarlos de dos maneras diferentes, una a partir de las “Categorias Estándares” que predefine SPEM, y otra a partir de las “Categorías Personalizadas” que permite definir el metamodelo.
Es importante destacar, antes de continuar con la definición de las “Categorías”, que no se han incluido los “Elementos de Método” definidos en SPEM como “Guías”, ya que no han sido incluidos dentro del alcance de este Plugin.

Se definieron las “Categorías Estándares” (predefinidas por SPEM) de:
“Disciplinas”, realizando una clasificación de las Tareas, según su objetivo o sector de realización.

1) Tareas de análisis
2) Tareas de construcción
3) Tareas de corrección
4) Tareas de diseño
5) Tareas de documentación
6) Tareas de identificación
7) Tareas de incorporación
8) Tareas de prueba
9) Tareas de revisión
10) Tareas de validación
11) Tareas de verificación

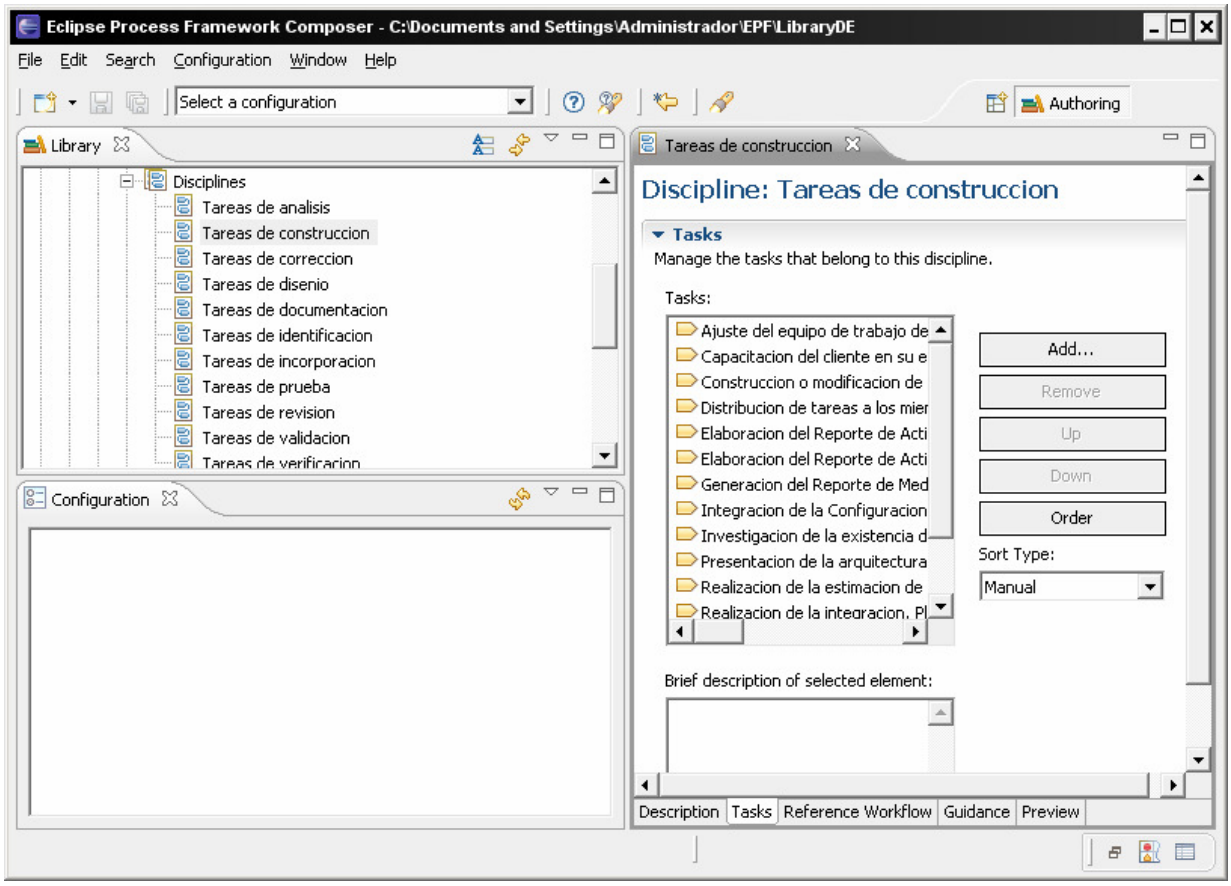

Ilustración 49. Disciplinas.

A su vez, se incluyó a cada Tarea dentro de su Categoría correspondiente:

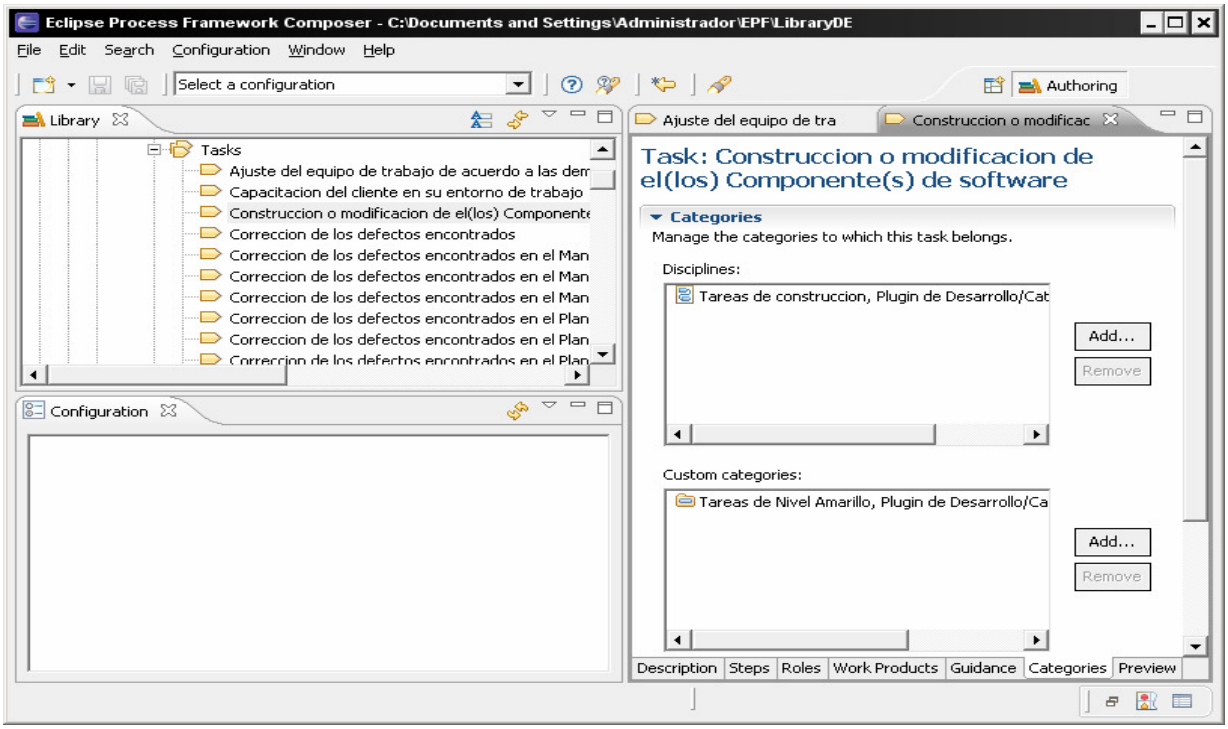

Ilustración 50. Categorías de Tareas.

"Dominios", clasificando los Productos de Trabajo según su tipo.

1) Casos de Prueba
2) Documentos
3) Manuales
4) Planes
5) Productos de fase de Requerimientos
6) Reportes

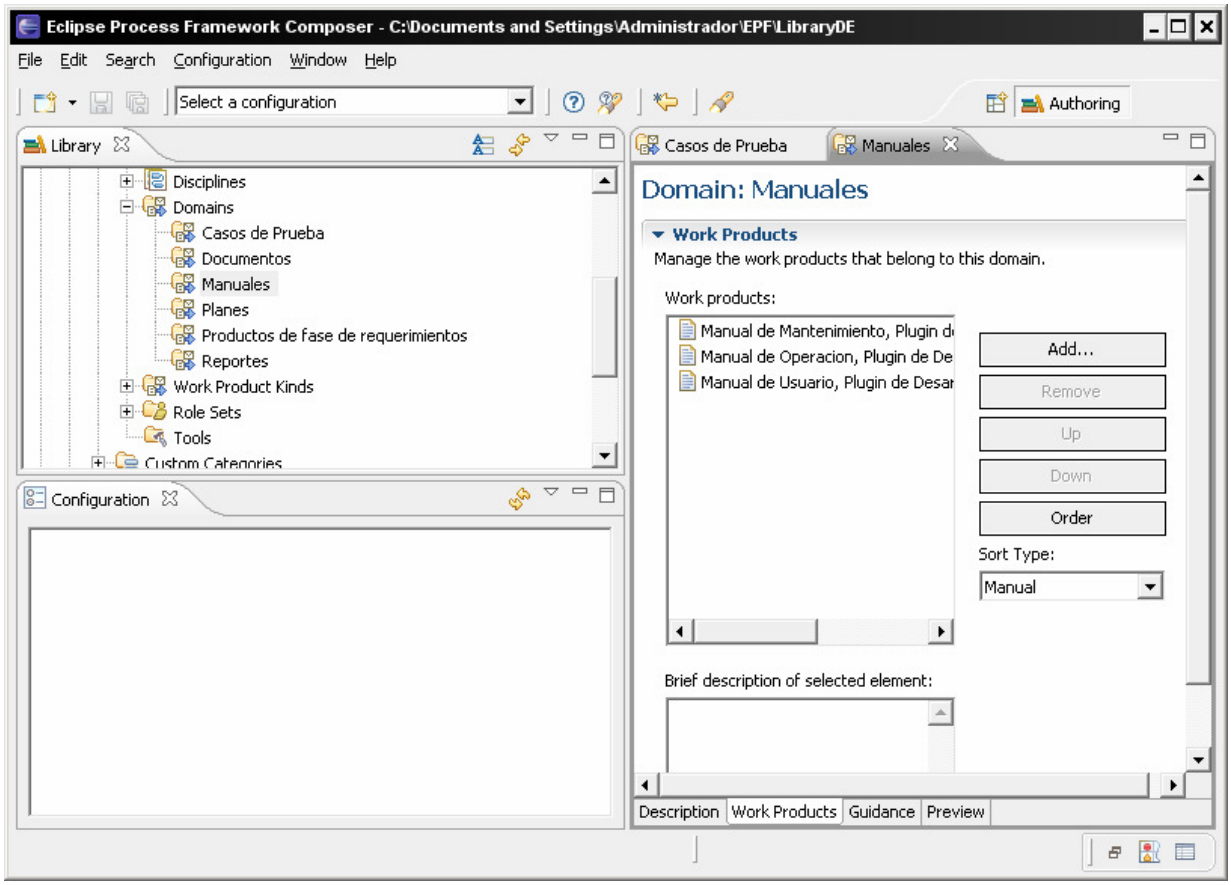

Ilustración 51. Dominios.

También se asoció a cada "Producto de Trabajo" con su Categoría correspondiente:

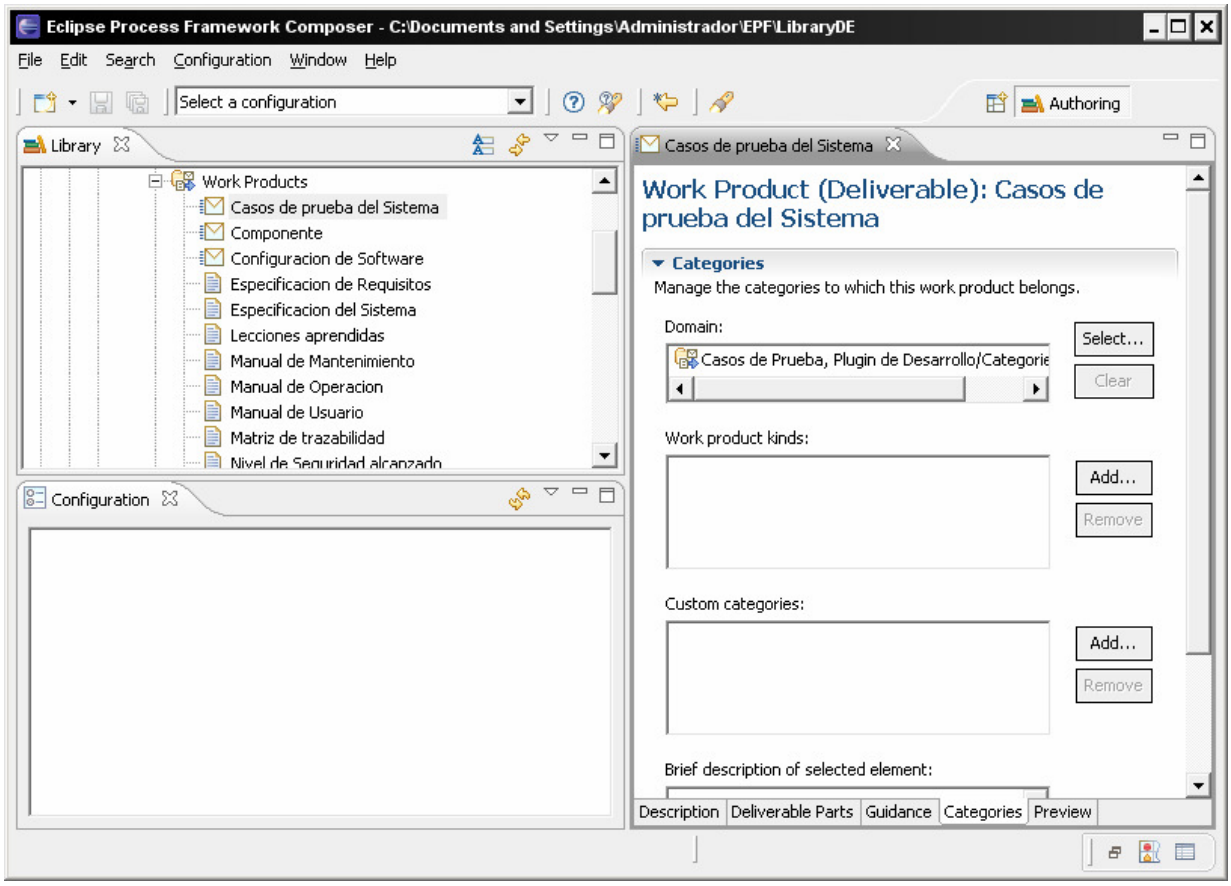

Ilustración 52. Categorías de Producto de Trabajo.

Conjunto de roles, clasificados de acuerdo a la actividad desempeñada por cada Rol.

1) Analistas
2) Arquitectos
3) Diseñadores
4) Responsables

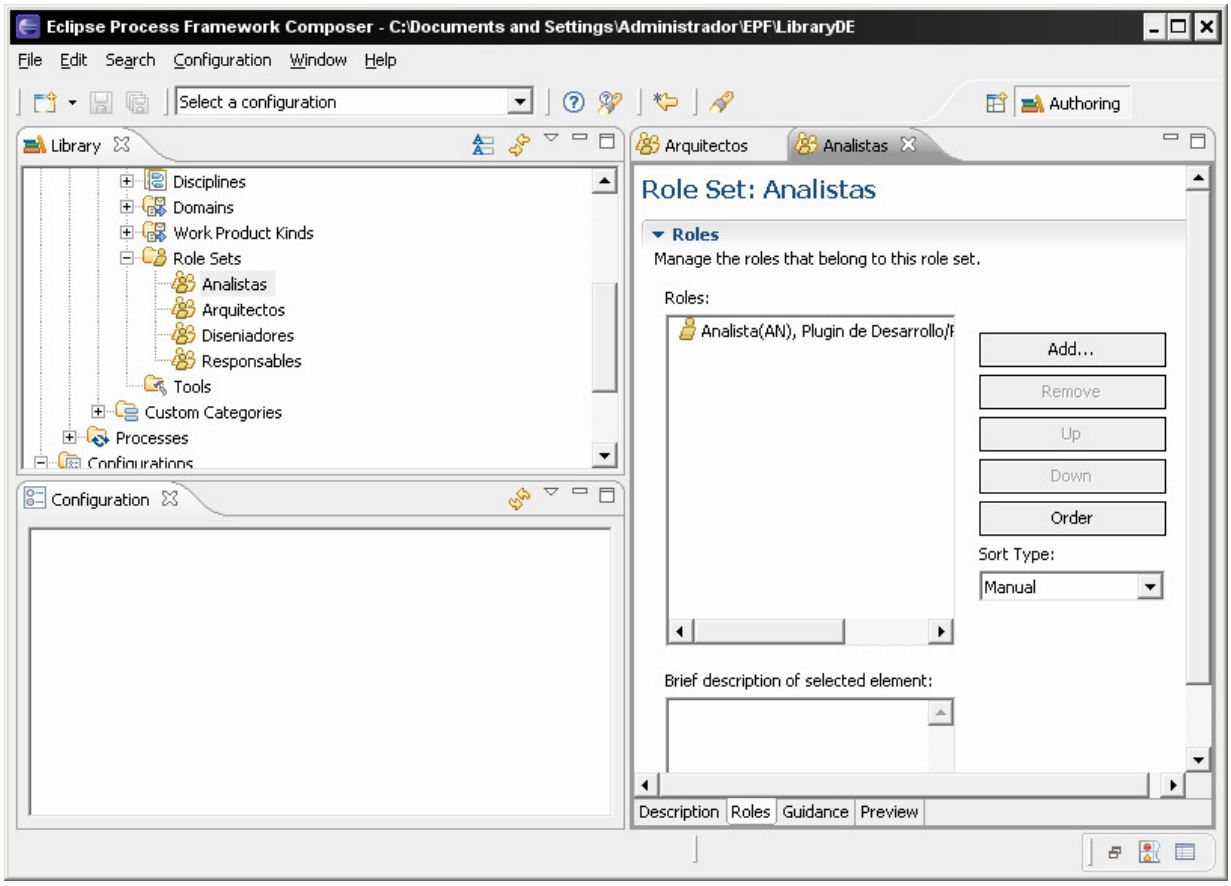

Ilustración 53. Conjunto de Roles.

Asociando también, a cada Rol definido en los Contenidos de Método, su Categoría correspondiente.

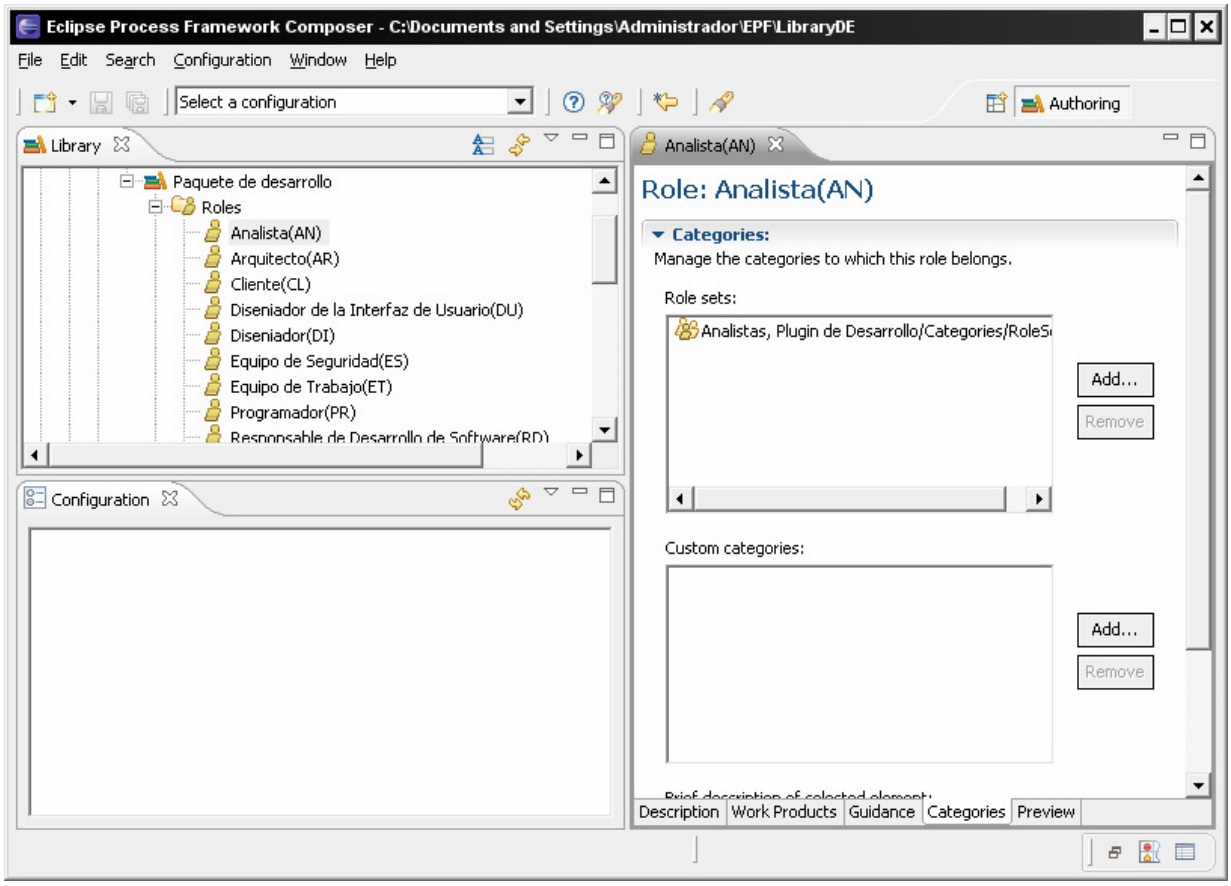

Ilustración 54. Categoría de Rol.

Las “Categorías Estándares”, se definieron sin seguir el modelo de Proceso de COMPETISOFT. Es decir, en la clasificación de las Tareas, se tuvo en cuenta el área o tema con el que se relaciona la misma, al igual que cuando se clasificaron los Roles o los Productos de Trabajo.

También se definieron las “Categorías Personalizadas” (éstas, a diferencia de las estándares, que ya vienen predefinidas en SPEM, se declararon como nuevas categorías del modelo). Esto se hizo en base al nivel de madurez de cada tarea. Se declaró una categoría de tareas según nivel de madurez de proceso al que pertenecen.

Los niveles en COMPETISOFT están definidos de acuerdo a colores que los identifican, para definir estas categorías se siguió este patrón.

Tareas según nivel de madurez:
-Tareas de Nivel Amarillo
-Tareas de Nivel Azul
-Tareas de Nivel Verde
-Tareas de Nivel Rosa

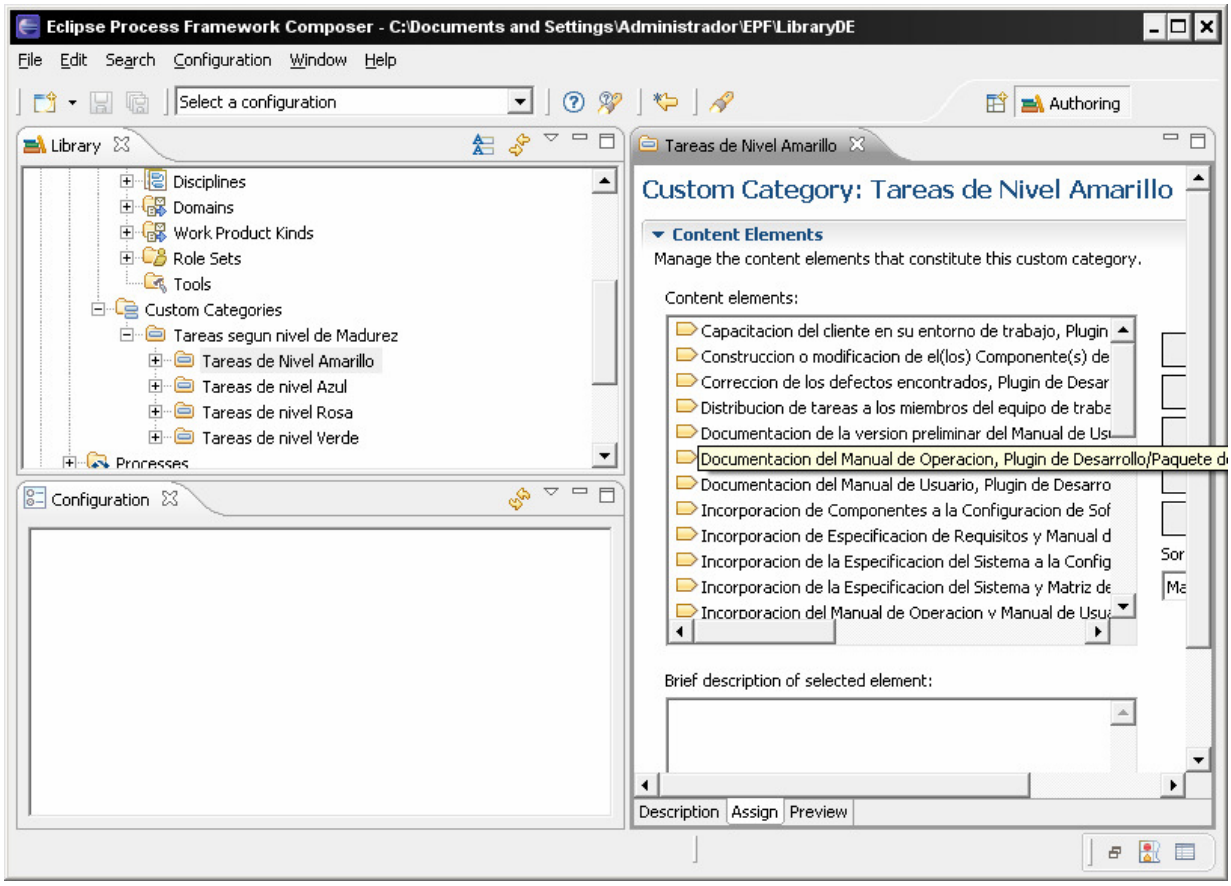

Ilustración 55. Categorías Personalizadas.

Como siguiente paso se definieron los "Procesos", de una forma ascendente, facilitando la reutilización de todos los elementos definidos previamente en la sección de "Contenidos de Métodos"

Se definieron "Patrones de Capacidad", según una clasificación hecha por el nivel de capacidad en el que se encuentran definidos en COMPETISOFT.

Es decir se definieron cuatro patrones de Procesos:

- PROCESO DE DESARROLLO DE SOFTWARE EN COMPETISOFT NIVEL AMARILLO
- PROCESO DE DESARROLLO DE SOFTWARE EN COMPETISOFT NIVEL AZUL
- PROCESO DE DESARROLLO DE SOFTWARE EN COMPETISOFT NIVEL VERDE
- PROCESO DE DESARROLLO DE SOFTWARE EN COMPETISOFT NIVEL ROSA.

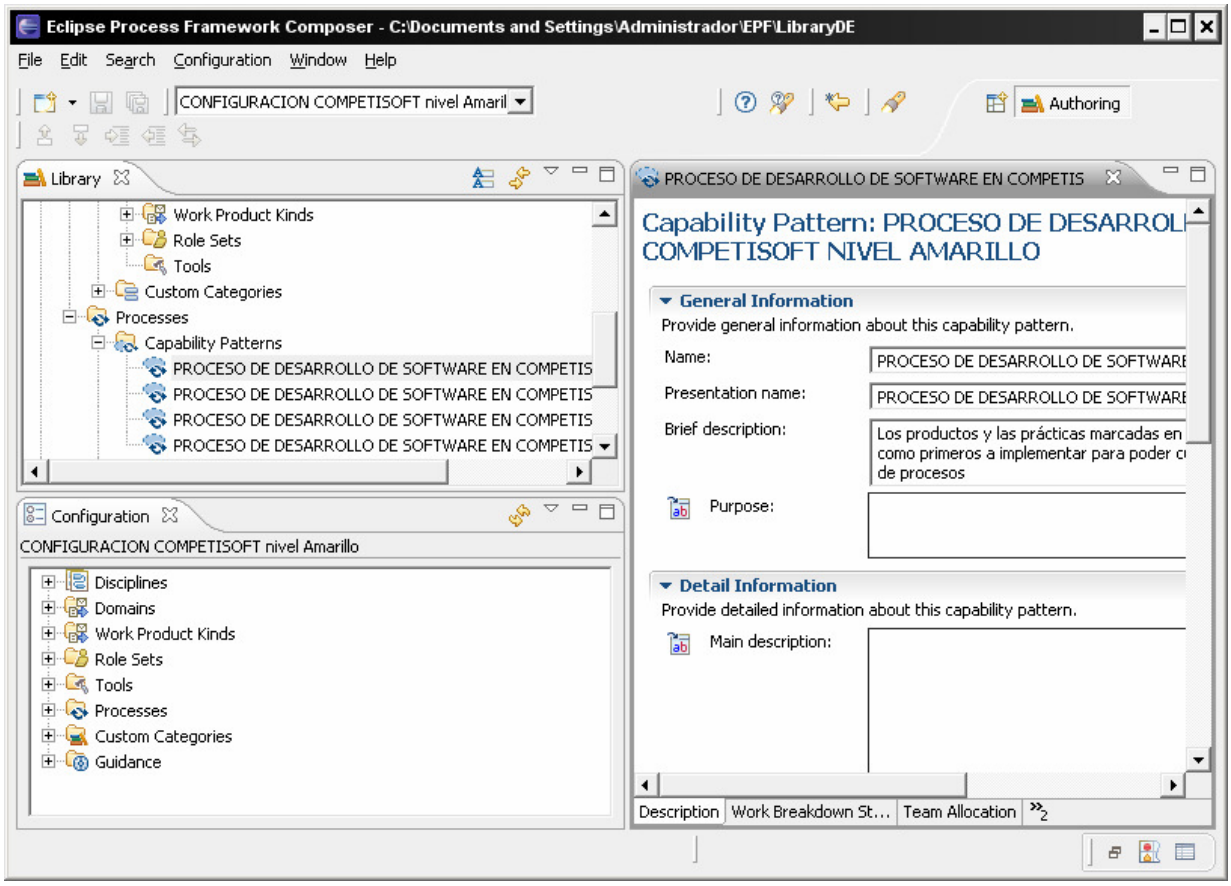

Ilustración 56. Procesos.

En el PROCESO DE DESARROLLO DE SOFTWARE EN COMPETISOFT NIVEL AMARILLO, por ejemplo, se definió su estructura de desglose de trabajo (concepto de SPEM) “Work Breakdown Structure”, instanciando 8 actividades que se corresponden con las 8 fases de desarrollo que se definen en el modelo de COMPETISOFT:

- Realización de la Fase de Inicio
- Realización de la Fase de Requisitos
- Realización de la Fase de Análisis
- Realización de la Fase de Diseño
- Realización de la Fase de Construcción
- Realización de la Fase de Integración
- Realización de la Fase de Prueba
- Realización de la Fase de Cierre

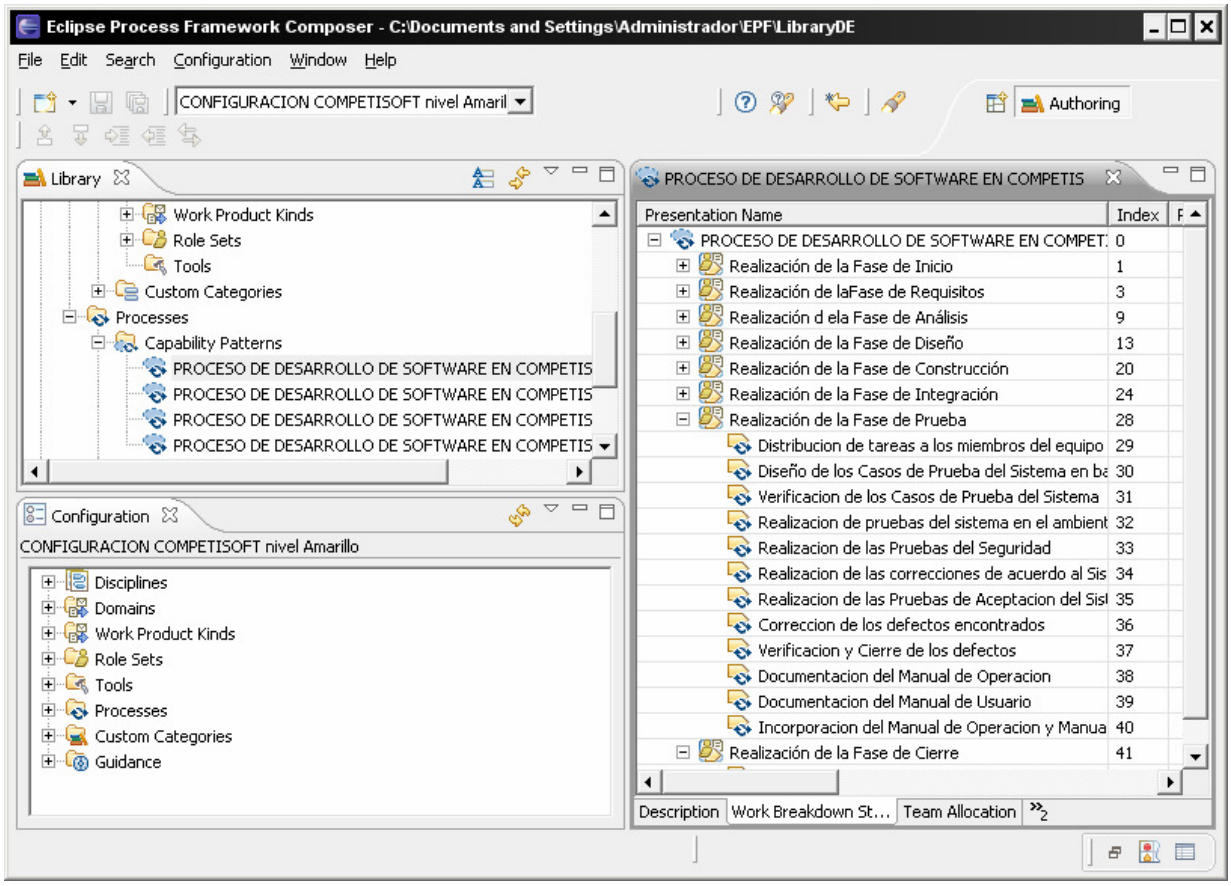

Ilustración 57. Work Breakdown Structure.

Dentro de cada una de estas actividades (“Activity”), se definieron los “Task Descriptor” (las tareas definidas en los Contenidos de Métodos) que las componen.

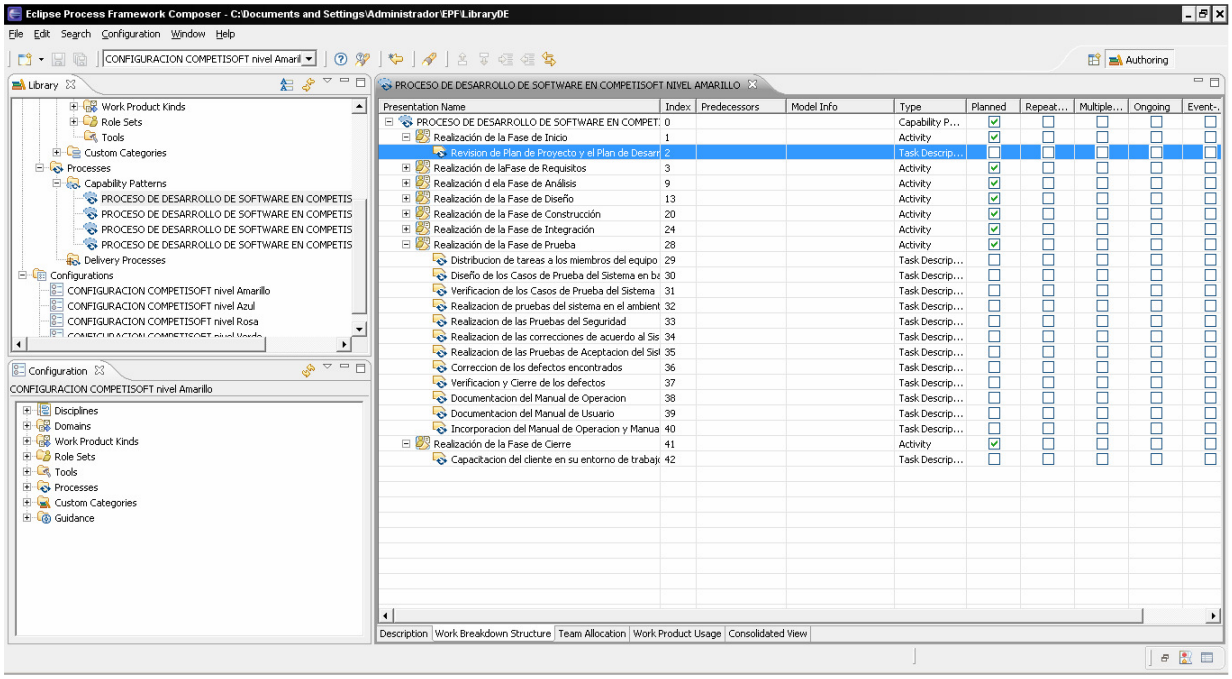

Ilustración 58. Activity.

También se definieron los “Work Product Usage” de cada “Activity”, que corresponde a la definición de los Productos de Trabajo asociados a cada actividad. Y los “Team Allocation”, asociando los Roles responsables de cada actividad.

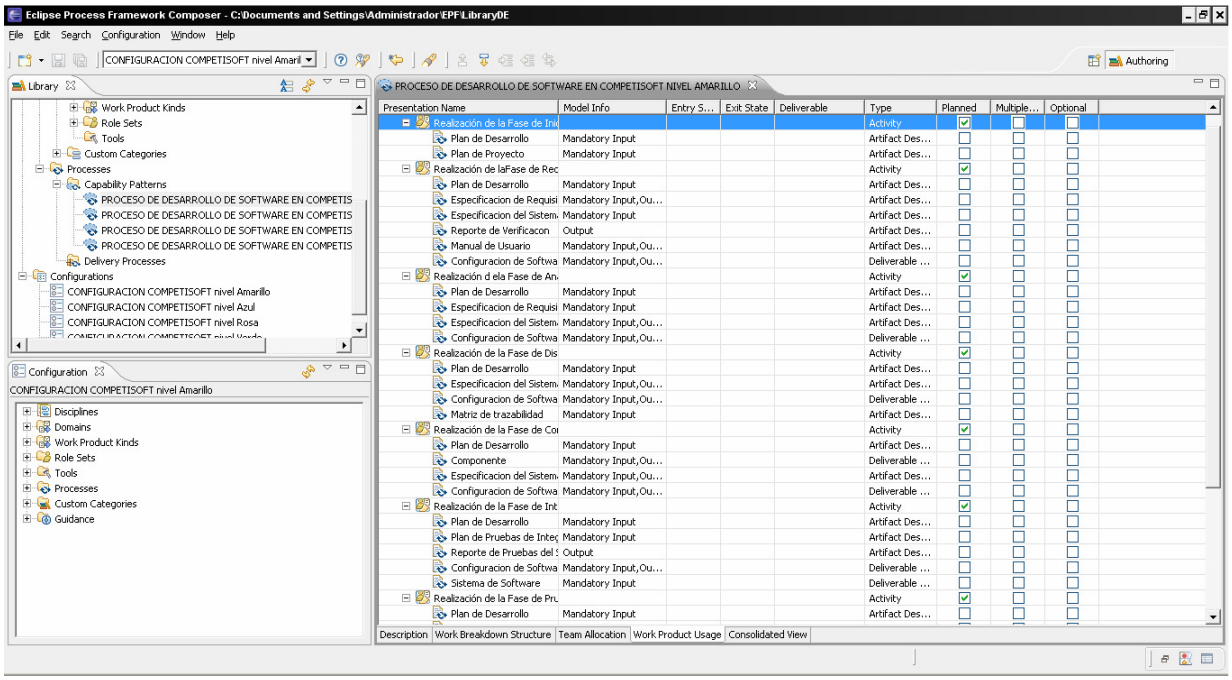

Ilustración 59. Work Product Usage.

Es importante destacar que el alcance de este Plugin fue la instanciación de “Patrones de Procesos” sencillos, dejando fuera del mismo la posibilidad que ofrece SPEM y la herramiento EPFC de definir “Patrones de Procesos” más complejos, que reutilicen los más sencillos y se obtegan “Procesos para Despliegue”.

A continuación se definieron las “Configuraciones de Métodos”, obteniendo así una Configuración de nivel amarillo, una azul, una verde y una rosa, cada una asociada a un “Proceso” diferente.

Las “Configuraciones de Método” (“Method Configuration”), se instanciaron seleccionando contenidos de los Plugins de la Biblioteca de Métodos (“Method Library”).

Esto se realizó seleccionando para cada “Configuración” definida, los “Contenidos de Método” correspondientes a cada “Proceso” definido por nivel de Madurez, para así obtener una configuración para cada nivel.

Es preciso destacar que se podría realizar una configuración que tome “Elementos de Método” de diferentes “Procesos”, para así obteber nuevas vistas de los estos y sus combinaciones.

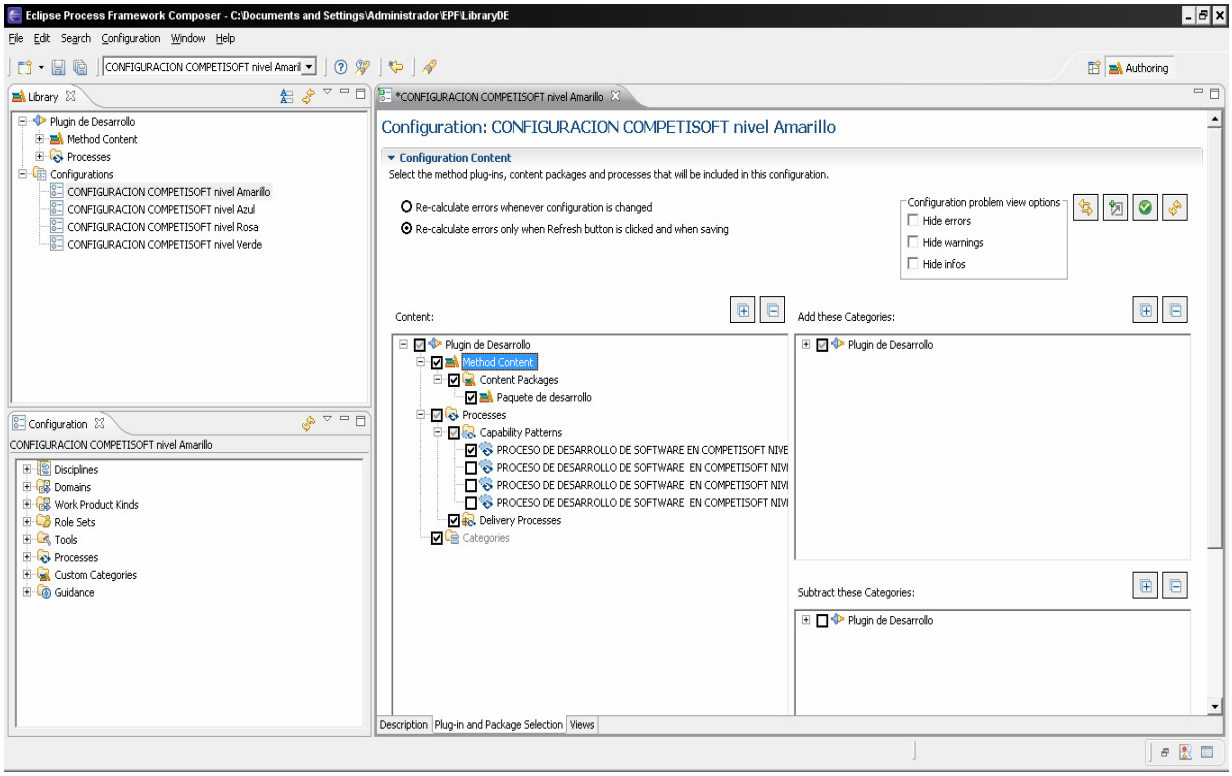

Ilustración 60. Method Configuration.

Por último y utilizando las configuraciones ya definidas, se publicó el "Proceso de Desarrollo de nivel Amarillo" a modo de ejemplo de una configuración publicada en la Web. Es necesario aclarar, que en el Plugin, no se colocaron acentos, ni letras 'ñ', en las palabras, para evitar errores y advertencias, al generar la página Web.

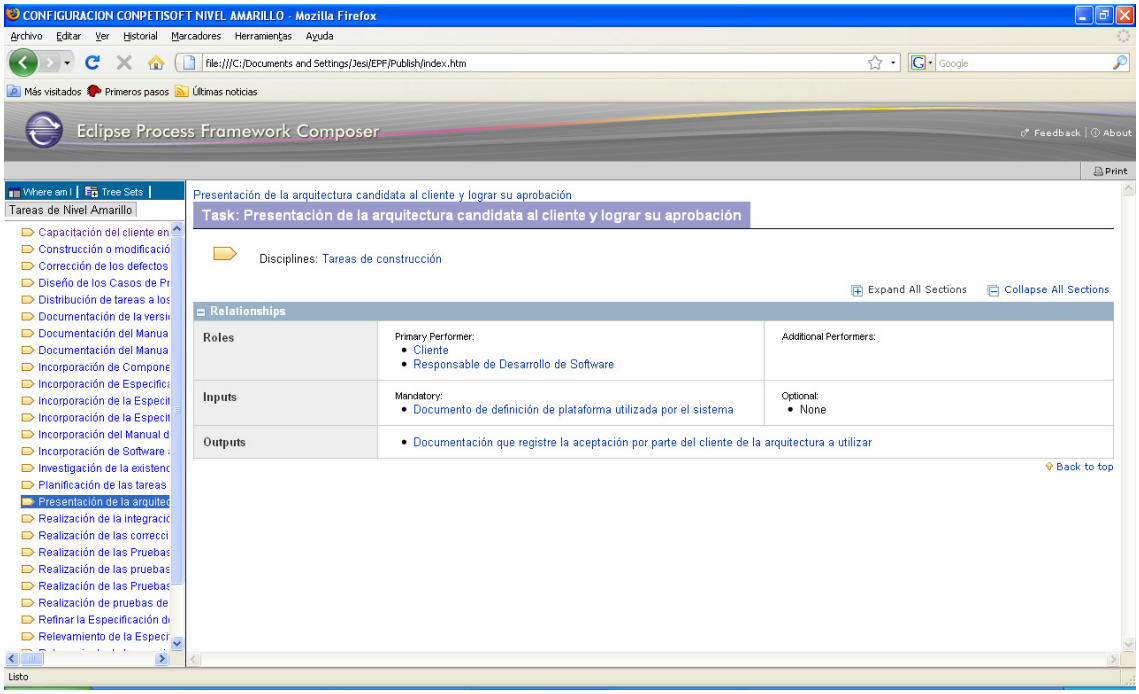

Ilustración 61. Configuración publicada en la Web.

5. Resultados esperados

A partir de esta investigación se pretende obtener un Plugin de Desarrollo basado en el proceso definido en COMPETISOFT, utilizando el metamodelo SPEM y la herramienta EPFC. Para así brindar un modelo que las organizaciones desarrolladoras de Software puedan adaptar a sus proyectos.

Se pretende dar conocimiento de la importancia de una técnica de modelado de proceso que permita coordinar los tiempos, respetar costos y lograr la funcionalidad deseada en plazos estimados, para obtener productos de calidad.

Por otro lado, también se espera poder brindar la información recopilada en la investigación y el estudio realizado sobre SPEM y el uso de la herramienta EPFC, así como también dar la posibilidad del surgimiento de dudas e iniciativa para investigar nuevos temas relacionados a estos.

6. Trabajos futuros

Durante el desarrollo de esta tesis se investigó el uso de la herramienta EPFC, se analizó el metamodelo de SPEM y todos sus posibles usos. También se analizó el Modelo de Procesos de COMPETISOFT centrándose en el Proceso de Desarrollo definido por el mismo.

Como futura investigación, se propone estudiar los restantes procesos definidos en COMPETISOFT a nivel de Categoría de Operación, como lo son el de Administración de un Proyecto Específico y Mantenimiento de Software, desarrollar un Plugin que los represente así como se hizo con el Proceso de Desarrollo, analizar también los diferentes niveles de capacidad de sus actividades y representarlo de forma de ser adaptado a las necesidades de cada organización.

Con este nuevo desarrollo se estaría obteniendo una representación completa del modelo de procesos de COMPETISOFT, así como también una publicación Web general de todo el proceso.

Ampliar este Plugin para obtener una visión general y completa de todo el modelo de procesos de COMPETISOFT abarcando todas las áreas y definiciones posibles.

7. Conclusiones

El modelado de un Proceso de Software es tan importante como su desarrollo. Modelarlo permite lograr la madurez del Proceso de Software, es decir poder definir, documentar, medir y mejorar continuamente.

El modelo de un Proceso de Software debe ser preciso, para poder gestionar el proceso de forma correcta, obteniendo un producto confiable y fácil de mantener, pudiendo realizar cambios en un futuro o durante el desarrollo, sin la necesidad de la redefinición total del Proceso.

Cuando se desarrolla un producto de Software el objetivo a alcanzar es obtener calidad en el producto y ésta se consigue si se implementa un Proceso con calidad.

Es por esto que se implementó este Plugin, que brinda la posibilidad de adaptación para cada proyecto particular según las necesidades y prioridades que se consideren más relevantes. Se utilizó y estudió el Proceso de Desarrollo implementado en COMPETISOFT, ya que se consideró interesante, por ser un modelo aplicable a las pequeñas y medianas empresa, y fácil de comprender y aplicar como un modelo de referencia. Además se eligió el metamodelo SPEM v.2.0 por la gran cantidad de ventajas que ofrece, al poder reutilizar componentes y brindar un repositorio donde almacenar los procesos, permitiendo su edición y recuperación de forma clara y fácil.

A partir del Proceso de Desarrollo definido en COMPETISOFT, se estudió cómo mapear las actividades allí definidas, haciendo uso de las posibilidades que brinda la herramienta EPFC de representación y clasificación de estos componentes.

Se definieron las entradas y salidas de cada actividad particular, que aparecen en el proceso como resultado de las tareas realizadas dentro de una actividad, esto permitió poder reflejar dependencia entre las actividades.

El Plugin permitirá contar con una aplicación estática, con posibilidad, incluso, de ser descargada de la Web, con el deseo de que las organizaciones puedan reutilizarla en sus proyectos. Gracias a él, se podrá también obtener diferentes vistas de la información según la configuración seleccionada para el trabajo, se posibilitará también generar otras instancias del mismo, con más o menos contenidos según el proceso a modelar.

Como principal objetivo se destaca la obtención de la instanciación de un Plugin de desarrollo con una configuración determinada, dada por el modelo propuesto por COMPETISOFT, y basada en los conceptos estudiados en SPEM.

La obtención del Plugin, fue útil para la comprensión de los conceptos definidos en SPEM, ya que al instanciar el Proceso de Desarrollo de COMPETISOFT, se pudieron concretar muchas de sus definiciones y analizar profundamente otras.

8. Agradecimientos

Quiero agradecer, en primer lugar a mi co-directora de tesis, Silvia Esponda, por su dedicación, predisposición y apoyo, en cada momento de la realización de esta Tesis.

Agradezco, su trabajo, su ayuda y paciencia durante todo el proceso de elaboración. También, agradezco a mi directora, Patricia Pesado, por su colaboración y tiempo prestado.

Además, quisiera aprovechar este espacio, para agradecer en forma más general, a todas aquellas personas que formaron parte de esta realización, directa o indirectamente, brindándome su apoyo tanto académico como espiritual.

Muchas personas colaboraron conmigo, para poder concretar esta carrera y estoy segura que sin ellos nada hubiese sido posible.

Quiero agradecer a mi familia, por estar siempre conmigo, apoyándome en todo, brindándome ese afecto incondicional, que fue sumamente importante para adquirir las fuerzas y la voluntad para seguir adelante. El estar fisicamente lejos de ellos, me ayudo a valorar más el tenerlos cerca.

Agradecer a mi "compañero de vida" de estos años, que me brindó su amor, para hacer que las cosas, a la distancia, no fueran tan duras y todo resultara más fácil.

Me gustaría darle las gracias, a mis compañeros de facultad, "mis amigos", con los que he compartido el esfuerzo, y la lucha por llegar al final de esta carrera, que sin ellos no hubiese podido realizar.

También agradecerle a mis compañeros de la casa de estudiantes de San Pedro, con los que compartí todos estos años y a los que voy a recordar siempre.

Gracias a todos, porque sin ustedes, nada hubiese sido posible. Gracias por permitirme finalizar esta etapa de mi vida, y haber hecho posible su disfrute y compañía.

Hoy estoy feliz de concretarla y estoy feliz de sentir que han compartido conmigo mucho, y que aún queda mucho por compartir.

9. Referencias

- [COM1] Contribución de los estándares internacionales a la gestión de Procesos de Software. Francisco J. Pino 1,2, Félix García 2, Mario Piattini. Grupo IDIS Facultad de Ingeniería Electrónica y Telecomunicaciones. Universidad del Cauca. Campus Universitario, Ronda de Toledo s/n, Ciudad Real, España. ISSN: 1698-2029 RPM-AEMES, VOL. 4, Num. 2, Abril 2007.

- [COMPE01] COMPETISOFT Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica Perfil 1 Administración de un proyecto Específico, Desarrollo de Software y Mantenimiento de Software. Elaborado con base en COMPETISOFT_v04. Piattini Velthuis. Grupo Alarcos Universidad de Castilla-La Mancha Versión 0.3 Agosto 2007

- [COMPE02] III Semana del CMMI. COMPETISOFT: Mejora de procesos para PyMEs. Mario Piattini Velthuis, Grupo Alarcos. Universidad de Castilla-La Mancha.

- [EPF01] Haumer, Peter. Overview to Eclipse Process Framework Composer Part 1: Key Concepts. IBM Rational Software.2006

- [EPF02] Haumer, Peter. Overview to Eclipse Process Framework Composer Part 2: Authoring method content and processes. IBM Rational Software.2006

- [EPF03] Eclipse Process Framework Componer. Reporte Técnico, Grupo de Ingeniería de Software. PROYECTO MMEDUSA., David Esteban López B, Hincapié Londoño, Jesús Andrés. Universidad EAFIT. Medellín. 2007.

- [GES01] Gestión Integrada del Modelado y de la Medición del Proceso Software. Félix García, Francisco Ruiz, José Antonio Cruz, Mario Piattini. Grupo Alarcos. Escuela Superior de Informática. Paseo de la Universidad, 4 13071 Ciudad Real (España) 2008.

- [GUIA] Guía de Uso de SPEM 2 con EPF Composer. Universidad de Castilla-La Mancha. Escuela Superior de Informática. Departamento de Tecnologías y Sistemas de Información. Grupo Alarcos. Versión 3.0.1. Francisco Ruiz, Javier Verdugo. -abril-2008.

- [ING01] Ingeniería de Software. Diseño, construcción y mantenimiento de sistemas de software grandes. Septiembre 2003. México. Dr. Pedro Mejía Álvarez CINVESTAV-IPN.

- [INTI01] Desarrollo dirigido por modelos (MDD). Junio 2007. Rosario – INTI.

- [LEN01] Notaciones y lenguajes de procesos. Una visión global.
  Juan Diego Pérez, Dr. Amador Duran Toro. Lenguajes y sistemas informáticos.

- [MDA01] MDA Explained: The Model Driven Architecture(TM): Practice and Promise (Addison-Wesley Object Technology Series) (Paperback). Anneke kleppe, Jos Warmer, Wim Bast.2003.

- [MODEL01] Model Driven Architecture (MDA) Document number ormsc. Architecture Borrad ORMSC. 2001-07-01.

- [MOM01] MOMENT: A Formal Framework for Model managem MENT Artur Boronat. Universidad Politécnica de Valencia. Junio 2007.

- [OMG01] Meta Object Facility (MOF) Core Specification OMG Available Specification Version 2.0 formal/06-01-01

- [OMG02] Software & Systems Process Engineering Metamodel Specification. Version 2.0. Agosto 2007.

- [PRES01] Pressman, Roger S.: "Ingeniería del Software, un enfoque práctico", Sexta edición. ISBN: 84-481-3214-9. México 2005.

- [REU01] MDA: Reusabilidad Orientada al Negocio. Valerio Adrián Anacleto. Octubre2006.

- [REV] Aplicación del modelado de procesos en un curso de Ingeniería de Software. Gabriel Alberto García Mireles. Jaime Nunó 2680 Ampliación Hidalgo, 22880 Ensenada, Baja California, México.Josefina Rodríguez Jacobo.Departamento de Ciencias de la Computación.Centro de Investigación Científica y de Educación.Superior de Ensenada (CICESE).Km. 107, Carretera Tijuana-Ensenada, 22830.Ensenada, Baja California, México. Revista Electrónica de Investigación Educativa Vol. 3, Nro. 2, 2001

- [SOM01] Sommerville, Ian: "Software Engineering", Fourth edition, Addison. Wesley, 1992.

- [UNI01] Fundamentos de SPEM 2.0 y EPF. Universidad de Castilla, La Mancha. Escuela Superior de Informática, Departamento de Tecnologías y Sistemas de Información. Ruiz, Francisco.

- [UNI02] Estudio de herramientas de desarrollo de software basado en modelos: MDA y Factorías de Software. Universidad Politécnica de Cartagena. Escuela Superior de Ingeniería de Telecomunicaciones Ramón García Soto. Director: José Juan Sánchez Manzanares. Septiembre/2006.

- [UNI03] Ensayo breve MDA Model Driven Architecture. Universidad Tecnológica de Pereira. Juan Pablo Gómez. 09/12/2007.

- [VIC01] Proyecto informático. Ingeniería de modelos con MDA. Estudio comparativo de Optimal J y ArcStyler. Departamento de Informática y Sistemas. Jesús Rodríguez Vicente. Junio 2004.

- [VUL01] Vulcano. Promoción del desarrollo de SW libre en un entorno de calidad y confianza adaptando las metodologías, procesos, modelos de negocio y últimas

tecnologías. FIT-350503-2007-7. Xabier Larrucea, Juncal Alonso, García Laura, Miguel Ángel. Barcelona.2007.

- [WEB01]http://www.tecnologiavirtual.com/index.php?option=com_content&view=article&id=71&Itemid=110.tecnologiavirtual.

- [WEB02] Ingeniería de Procesos Software. Aplicación a los Proyectos de Software. Planificación y Gestión de Sistemas de Información.
  Departamento de Tecnologías y Sistemas de Información.
  Escuela Superior de Informática. Universidad de Castilla-La Mancha.
  http://alarcos.esi.uclm.es/per/tmartinez/presIngProcesos.pps

- [WEB03] Hacia la Cuarta Generación del Software. Domingo. Jorge Ubeda. http://cuartageneracion.blogspot.com. Abril 18, 2006.

- [WEB04] Procesos de Software. Edición. Mara Ruvalcaba. http://www.sg.com.mx/content/view/4/11/. Enero-Febrero 2005.

- [WOR01] Workflow. Maestría en Ingeniería de Software. Módulo de Workflow. Universidad Nacional de La Rioja. Departamento de Ciencias Exactas Físicas y Naturales. UNLaR. Martínez Marcelo.