SPEM: Software Process Engineering Metamodel

Víctor Hugo Menéndez Domínguez y María Enriqueta Castellanos Bolaños
Facultad de Matemáticas
Universidad Autónoma de Yucatán
Mérida, México
{mdoming, enriqueta.c}@uady.mx

***Resumen*—Todas las organizaciones involucradas en el desarrollo de software necesitan establecer, gestionar y soportar el trabajo de desarrollo. El término "proceso de desarrollo de software" tiende a unificar todas las actividades y prácticas que cubren esas necesidades. Modelar el proceso de software es una forma para mejorar el desarrollo y la calidad de las aplicaciones resultantes. De entre todos los lenguajes existentes para el modelado de procesos, aquellos basados en productos de trabajo son los más adecuados. Uno de tales lenguajes es SPEM (Software Process Engineering Metamodel). SPEM fue creado por OMG (Object Management Group) como un estándar de alto nivel, que está basado en MOF (MetaObject Facility) y es un metamodelo UML (Uniform Model Language). Constituye un tipo de ontología de procesos de desarrollo de software. En este artículo se ofrece una descripción, en términos generales, del estándar SPEM. También se destacan los cambios que ha experimentado entre la versión 1.1 y la versión 2.0, presentando tanto las ventajas como las desventajas encontradas entre ambas versiones.**

***Palabras Clave*—Proceso de desarrollo de software, SPEM, MOF, UML.**

I. Introducción

Esta sección describe qué es un proceso de software y su relación con el estándar SPEM, también presenta la relación de SPEM con otros estándares definidos por OMG como UML, MOF y XMI.

A. Procesos de Software y SPEM

Todas las organizaciones involucradas en el desarrollo de software necesitan establecer, gestionar y soportar el trabajo de desarrollo. El término "proceso de desarrollo de software" tiende a unificar todas las actividades y prácticas que cubren esas necesidades [10].

El principal objetivo de tal proceso es garantizar la construcción de un software adecuado, conforme a sus especificaciones y dentro de los límites de tiempo y costo [2]. Sin embargo, es un proceso complejo conformado de múltiples actividades que son interdependientes, incluyendo el desarrollo técnico, la gestión de proyectos, el control de calidad y las actividades de soporte del cliente [17].

Modelar el proceso de software es una forma para mejorar el desarrollo y la calidad de las aplicaciones resultantes [2]. La Ingeniería de Procesos de Software (SPE, Software Process Engineering) consiste en modelar, diseñar, mejorar y aplicar procesos utilizando lenguajes de modelado de procesos (PML, Process Modelling Language) [10]. De todos los lenguajes para el modelado de procesos existentes, aquellos basados en productos de trabajo son los más adecuados para modelar los procesos de desarrollo de software [11].

Uno de tales lenguajes es SPEM (Software Process Engineering Metamodel) [12], una especificación de OMG (Object Management Group) que está basado en MOF (MetaObject Facility) y es un metamodelo UML (Uniform Model Language). Al ser SPEM un metamodelo UML, utiliza su notación, lo que permite visualizar, especificar, construir y documentar sistemas orientados a objetos [5].

SPEM permite representar una familia de procesos de desarrollo de software y sus componentes. Constituye un tipo de ontología de procesos de desarrollo de software [18]. Para ello provee un conjunto de elementos de modelado de procesos para describir cualquier proceso de desarrollo de software sin agregar modelos o restricciones de alguna área o disciplina específica, tal como gestión de proyectos o análisis.

SPEM proporciona una sintaxis y estructura para cada aspecto de los procesos desarrollo, incluyendo:

- Roles.
- Tareas.
- Artefactos.
- Lista de verificación
- Productos de trabajo.
- Técnicas y herramientas.
- Estructuras de trabajo.
- Capacidad de rastreo y refinamiento.
- Ayuda sensible al contexto, guía y lineamientos.
- Descripción textual de elementos.

B. MOF, SPEM y XMI

Mientras que el propósito de SPEM es convertirse en el lenguaje estándar para el modelado de procesos de software, MOF pretende ser el lenguaje universal de metamodelos, capaz de describir lenguajes tales como el mismo SPEM, UML, modelos relacionales, etc. MOF constituye una parte de la arquitectura estandarizada de cuatro niveles, propuesta por OMG, convencionalmente denominados M0 - M3 [6].

Un proceso de desarrollo – esto es un proceso de producción del mundo real – como se indica en la Figura 1, es el nivel M0. La definición del proceso correspondiente (Modelo) está en el nivel M1, por ejemplo cualquier proceso genérico como el RUP o XP, u otras especificaciones adaptadas a un proyecto dado. Sin embargo, SPEM se enfoca en el metamodelo que se ubica en el nivel M2 y sirve como plantilla para el nivel M1. Con el fin de evitar un número infinito de niveles, OMG estableció hacer a M3 reflexivo. La consecuencia es que MOF debe ser capaz de describirse a si mismo [6].

XMI (XML Metadata Interchange) es la tecnología adoptada por la OMG para intercambiar modelos en forma serializada. XMI especifica cómo crear esquemas XML de modelos y se enfoca en el intercambio de metadatos de MOF, metadatos acorde a un metamodelo MOF [9].

XMI puede ser usado para manipular el metamodelo de SPEM de las siguientes formas [12]:

- Para crear definición de tipos de documentos SPEM.
- Para transferir modelos de procesos basados en SPEM como documentos XML, o describiendo el modelo como una instancia directa de SPEM (Uso del DTD SPEM) o

describiendo éste como un modelo de UML conforme al perfil UML para SPEM (Uso del DTD UML).

- Transformar el metamodelo de SPEM en un documento XML basado en el DTD MOF, para intercambio entre los repositorios conformes a MOF.

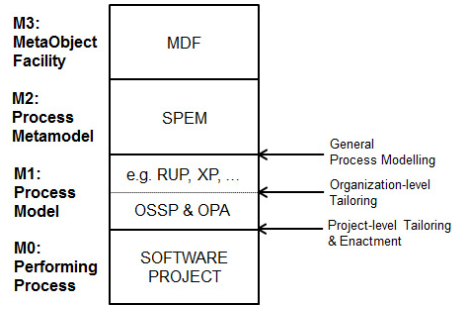


Fig. 1. Arquitectura estandarizada propuesta por OMG.

II. Estado del Arte

En esta sección se describe, en términos generales, el estándar SPEM, también se presentan los cambios que han surgido entre la versión 1.1 y la versión 2.0.

A. Software Process Engineering Metamodel, SPEM

SPEM fue creado por OMG como un estándar de alto nivel para describir procesos utilizados en el desarrollo de software orientado a objetos. Inicialmente fue creado como un metamodelo independiente pero más tarde fue reformulado para ser un perfil (profile) UML. Esto significa que se ajustaron los conceptos orientados a procesos para representar conceptos orientados al modelado [8].

En la base de SPEM está la idea de que un proceso de desarrollo de software es una colaboración entre las entidades activas abstractas llamadas Roles del proceso que realizan operaciones llamadas actividades en entidades concretas, tangibles llamadas productos de trabajo.

La Figura 2 describe un ejemplo del modelo conceptual fundamental usando la notación UML para una clase [9].

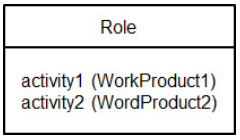


Fig. 2. Modelo conceptual SPEM.

Múltiples roles interactúan o colaboran para intercambiar productos de trabajo y provocar la ejecución, o representación de ciertas actividades. El objetivo principal de un proceso es traer unos productos de trabajo a un estado bien definido [10]. Para esto, un primer paso consistirá en reestructurar el rol, la actividad y el producto de trabajo. Esto nos conduce al modelo simple mostrado en la Figura 3.

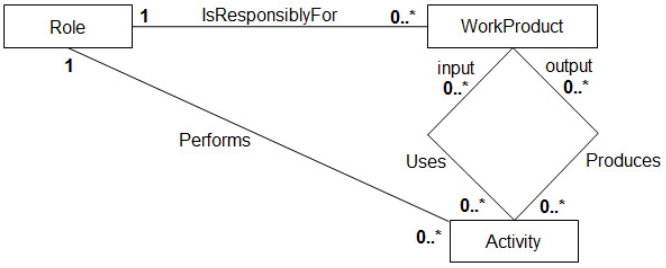


Fig. 3. Relación de los elementos fundamentales de SPEM 1.1 [12].

Las actividades utilizan productos de trabajo con el propósito de crear nuevos productos de trabajo. El trabajo es realizado usualmente por personas que juegan roles. Ellos realizan las actividades y son responsables de mantener directamente el conjunto de productos de trabajo [8].

La mayor unidad de trabajo es una actividad (Figura 4), la cual consta de un número de pasos. Estas clases describen el trabajo que se necesita realizar. El realizador de esta unidad de trabajo es el *ProcessRole*, el cual es un tipo de realizador de procesos, los cuales consisten en un conjunto de *WorkDefinition*. La gente interpreta roles para crear *WorkProduct* los cuales están asociados a *WorkDefinition* (y a actividades). Las definiciones de trabajo tienen precondiciones y poscondiciones (llamadas metas) como limitaciones.

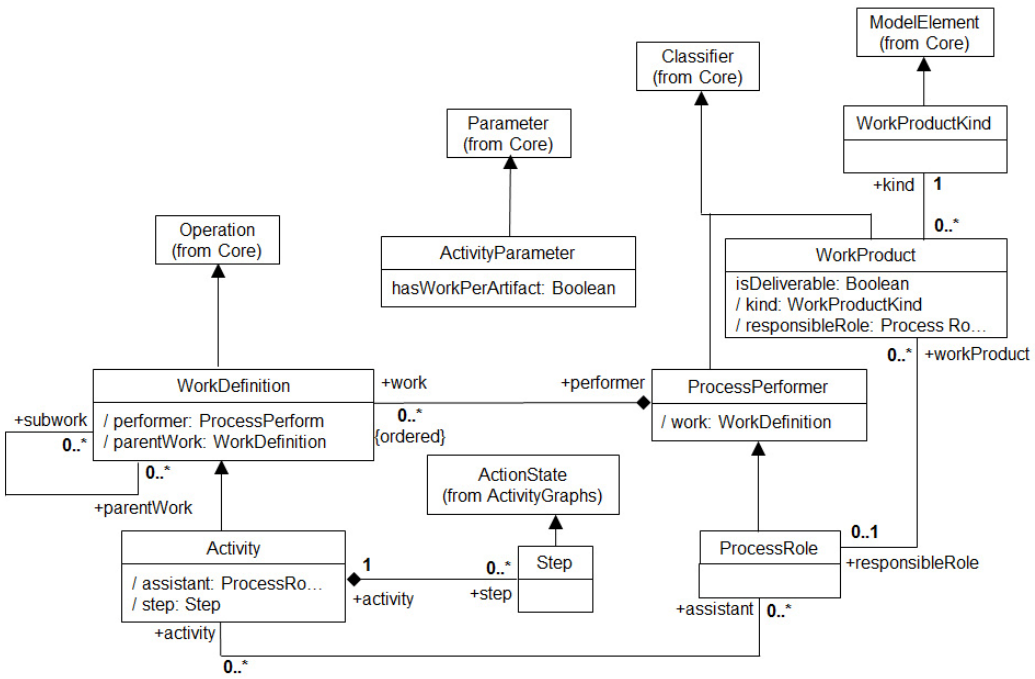


Fig. 4. Taxonomía de la actividad [12].

Un proceso se puede ver como una colaboración entre roles para alcanzar cierta meta u objetivo. Para dirigir su representación, podemos restringir la orden en la cual las actividades deben ser, o pueden ser ejecutadas. También es necesario definir la "forma" del proceso en un determinado tiempo, y la estructura del ciclo de vida en términos de fases y de iteraciones (Figura 5) [10]:

- *Phase* (*Fase*): es una especialización de *WorkDefinition* tal que su condición previa define los criterios de entrada de la fase y su meta (a menudo llamada *milestone*) define los criterios de la salida de la fase. Las fases son definidas con una restricción adicional de secuencialidad, que sus representaciones sean ejecutadas con una serie de fechas de tiempos de programación (*milestone*) que normalmente toman un mínimo (o ningún) traslape de sus actividades en el tiempo.
- *Lifecycle (Ciclo de vida)*: El ciclo de vida de un proceso es definido como una secuencia de fases que alcanzan una meta específica. Este define el comportamiento de un proceso completo que será representado en un proyecto o un programa dado.
- *Iteration (Iteración):* Una Iteración es un componente *WorkDefinition* que se caracteriza por tener el menor *milestone*.

SPEM también define elementos que son importantes para organizar otros elementos del proceso desde el punto de vista de la creación, ensamble y uso (Tabla I). Los paquetes están relacionados a la división de una o más descripciones de procesos en una parte autocontenida. Estos subelementos, denominados componentes de proceso, son porciones autocontenidas e internamente consistentes que pueden ser reutilizados con otros componentes de proceso para ensamblar un proceso completo. Pueden importar un conjunto no arbitrario de elementos de definición de procesos [10].

Los principales elementos de SPEM son los siguientes [12]:

- *WorkProduct* (*Producto de Trabajo*): es cualquier elemento producido, consumido, o modificado por un proceso. Esto puede ser una pieza de información, un documento, un modelo, código fuente y demás. Un *WorkProduct* describe una clase de producto de trabajo producido en un proceso.
- *WorkDefinition* (*Definición de trabajo*): es un tipo de operación que describe el trabajo que se ejecuta en un proceso. Este trabajo puede ser *Activity*, *Phase*, *Iteration* y *LifeCycle*.
- *Activity* (*Actividad*): es la principal subclase de *WorkDefinition*. Esta describe una parte del trabajo desarrollado por un *ProcessRole*: 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.
- *ProcessPerformer* (*Ejecutor de Proceso*): define la ejecución para un conjunto de definiciones de trabajo en un proceso. *ProcessPerformer* es una representación abstracta del representante del proceso completo o uno de sus componentes.
- *ProcessRol* (*Rol de Proceso*): define responsabilidades sobre productos de trabajo específicos y define los roles que ejecutan y asisten en actividades específicas.
- *ProcessPackage* (*Paquete de Procesos*): al igual que en UML, puede contener procesos propios e importar elementos de definición de procesos. Las actividades y definiciones de trabajo son contenidos respectivamente, por Roles de proceso y Ejecutores de proceso; Las máquinas de estado son contenidas por Productos de trabajo y sus propios estados y transiciones. Los grafos de actividad pueden ser contenidos por Paquetes, Clasificadores o características de comportamiento; Otros elementos de SPEM pueden ser contenidos por Paquetes.
- *Guidance* (*Guía*): pueden ser asociados con los elementos de SPEM, para proporcionar información más detallada a los ejecutores acerca de los elementos asociados. Los tipos de guía dependen de la familia de procesos y pueden ser por ejemplo: Guías, Técnicas, Métricas, Ejemplos, Perfiles UML, Tutoriales, Listas de comprobación, Plantillas, etcétera.
- *ProcessComponent* (*Componente de Proceso*): es una porción de descripción de proceso que es consistente internamente y puede ser reutilizado por otros *ProcessComponent* para ensamblar un proceso completo.

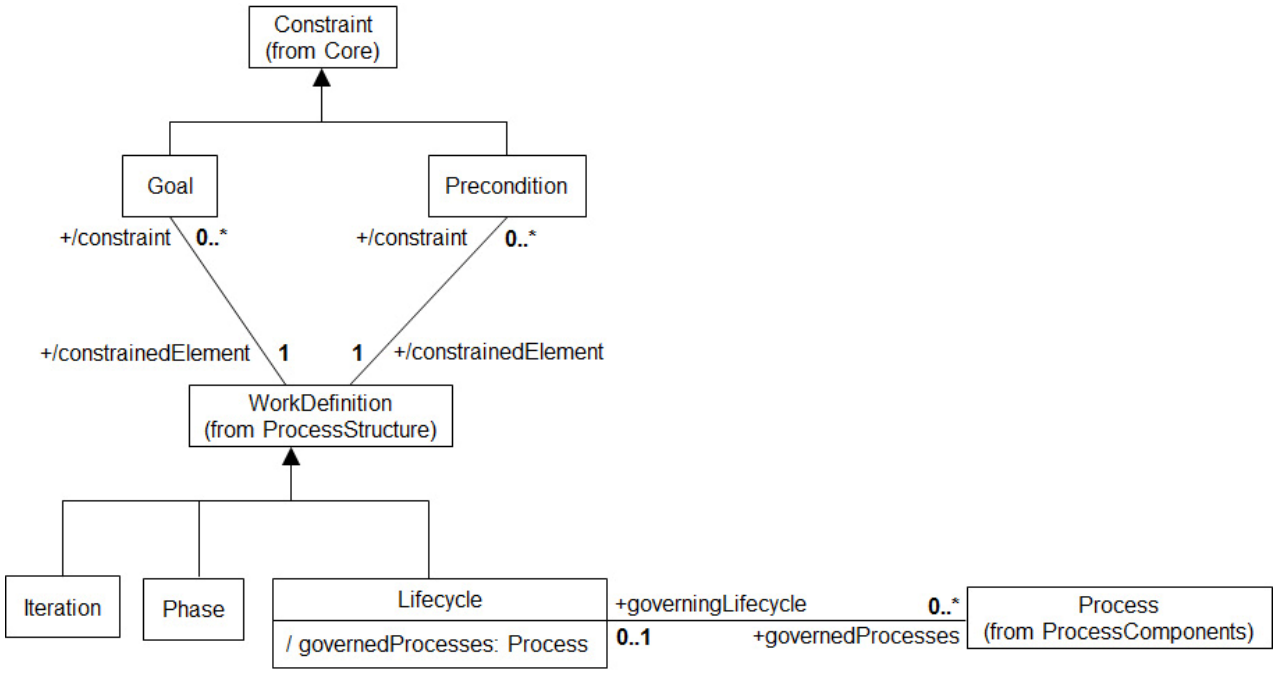


Fig. 5. Taxonomía del ciclo de vida [12].

- *Process* (*Proceso*): es un *ProcessComponent* que es capaz de mantenerse aislado de principio a fin. El proceso se distingue de un componente de proceso normal porque no está pensado para ser accedido por otros componentes.
- *Discipline* (*Disciplina*): es una especialización particular de Package que divide las actividades dentro de un proceso de acuerdo a un tema común. El particionamiento de actividades de esta forma implica que la guía asociada (*Guidance*) y la salida de *WorkProduct* se categoriza similarmente en un tema. La inclusión de una actividad en una Disciplina es representada por la dependencia de Categorías, con una restricción adicional de que toda Actividad es categorizada por exactamente una Disciplina.

TABLA I. ESTEREOTIPOS GRÁFICOS DE LOS ELEMENTOS SPEM

Elemento	Estereotipo UML
Activity	
Document	
Guide	
Process	
Phase	
Process Package	
Rol	
WorkProduct	
WorkDefinition	
Model	

Los diagramas UML pueden ser utilizados para presentar diferentes perspectivas de un modelo de procesos de software, siendo las más útiles:

- Diagramas de clase que permiten la representación de los siguientes aspectos de un proceso de software: herencia, dependencia, asociaciones, Comentarios y guías.
- Diagramas de paquete, que permiten la representación de procesos, componentes, paquetes y disciplinas. Las anidaciones son permitidas.
- Diagramas de caso de uso, que muestra las relaciones entre los roles del proceso y las definiciones del trabajo principal (Figura 6).
- Diagramas de secuencia, que ilustran los patrones de interacción entre las instancias del modelo SPEM.
- Diagramas de estado, que presentan el funcionamiento de los elementos del modelo SPEM. Se permite la anidación y el paralelismo pero no indicadores de historial.
- Diagramas de actividad, que describen la secuencia de actividades con sus productos de entrada y de salida, así como los estados de flujo (Figura 7).

Algunos diagramas y notaciones utilizados por UML han sido excluidos de SPEM y no tienen ningún significado.

B. Diferencias entre SPEM 1.1 y 2.0

A pesar de ser liberada desde 2002 y tener una revisión en el 2005 (SPEM 1.1), son pocas las implementaciones de SPEM 1.x en herramientas comerciales, y las que existen lo soportan de forma parcial. La causa de esta situación parece ser lo difícil que resultó su implementación, además que algunos elementos de la semántica son ambiguos [13]. La industria apenas está reconociendo la importancia de SPEM en el mercado de las herramientas metodológicas y técnicas.

La revisión de SPEM, pretende resolver esta situación. SPEM 2.0 es definido como un metamodelo así como un perfil de UML 2. SPEM 2.0 utiliza el mecanismo ofrecido por la infraestructura de UML para mezclar paquetes que gradualmente construyen el metamodelo. Proporciona unidades modulares que pueden ser usados como bloques de construcción para una especificación [13].

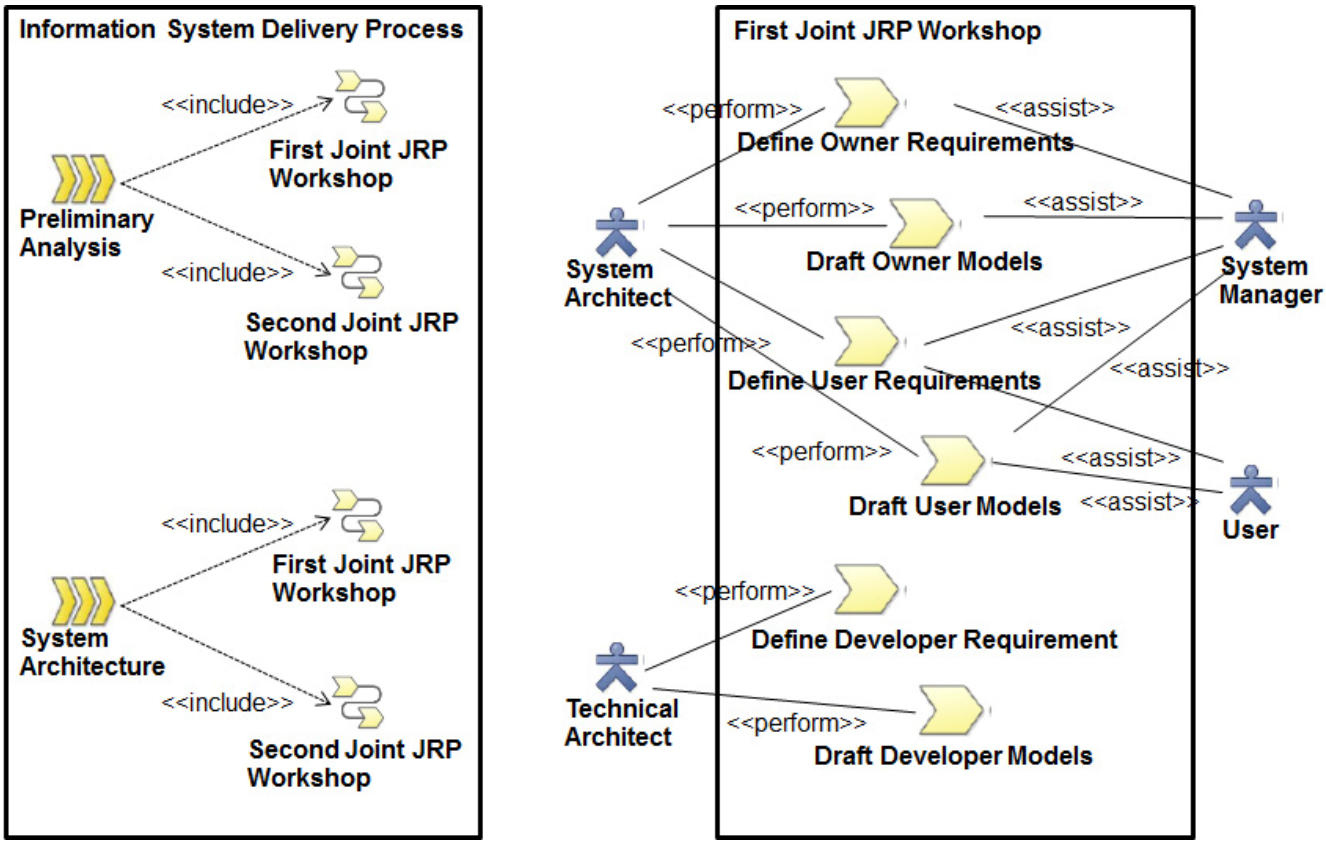


Fig. 6. Diagrama de caso de uso [12].

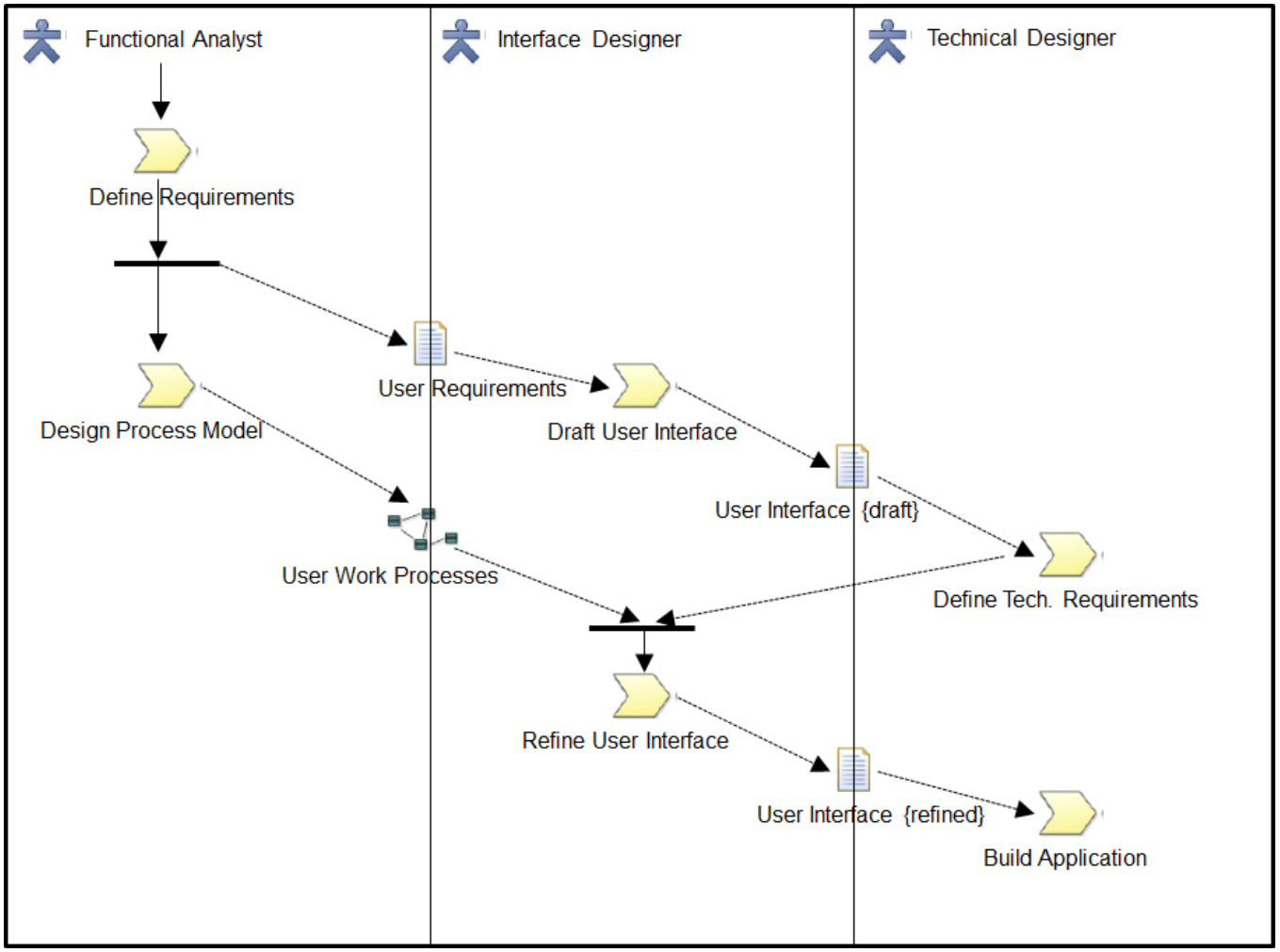


Fig. 7. Diagrama de actividades [12].

SPEM 2.0 está estructurado en siete paquetes de metamodelos principales. La estructura divide el modelo en unidades lógicas. Cada unidad extiende la unidad de la que depende, proporcionando estructuras y capacidades adicionales a los elementos definidos [13]:

1) *Core:* contiene las clases y abstracciones del metamodelo que constituyen la base para crear todas las clases de los demás paquetes.
2) *Process Structure:* Establece la base para todos los modelos de proceso y soporta la creación de estos en una forma fácil y flexible (Figura 8).
3) *Process Behavior:* Representa un proceso como una estructura estática, permitiendo anidar actividades y definir dependencias entre ellos.
4) *Managed Content:* Establece conceptos para la gestión de descripciones y contenido textual que apoyen a los conceptos de un proceso.
5) *Method Content:* Permite la construcción de una base de conocimientos del desarrollo definiendo conceptos del ciclo de vida y métodos, técnicas y las mejores prácticas.
6) *Process with Methods:* Define estructuras para integrar procesos definidos con *Process Structure* con conceptos de *Method Content*.
7) *Method Plugin:* Introduce conceptos para diseñar y gestionar repositorios de procesos reutilizables.

Las principales características y mejoras de SPEM 2.0 con respecto a sus versiones previas son [13]:

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

La Figura 9 presenta un diagrama de clases en SPEM 2.0, muestra que la tarea "Use Case Analysis" tiene entradas y salidas que son obligatorias y opcionales y que son desempeñadas por dos roles: "Designer", que es el rol principal y "System Analyst", que es opcional. Los estereotipos pueden tener representaciones gráficas y textuales. Para mantener compatibilidad con SPEM 1.1, la mayoría de los estereotipos tienen su contraparte (Figura 10), sin embargo, dado que SPEM 2.0 introduce nuevos conceptos, no todos tienen su equivalente en SPEM 1.1 (Tabla II).

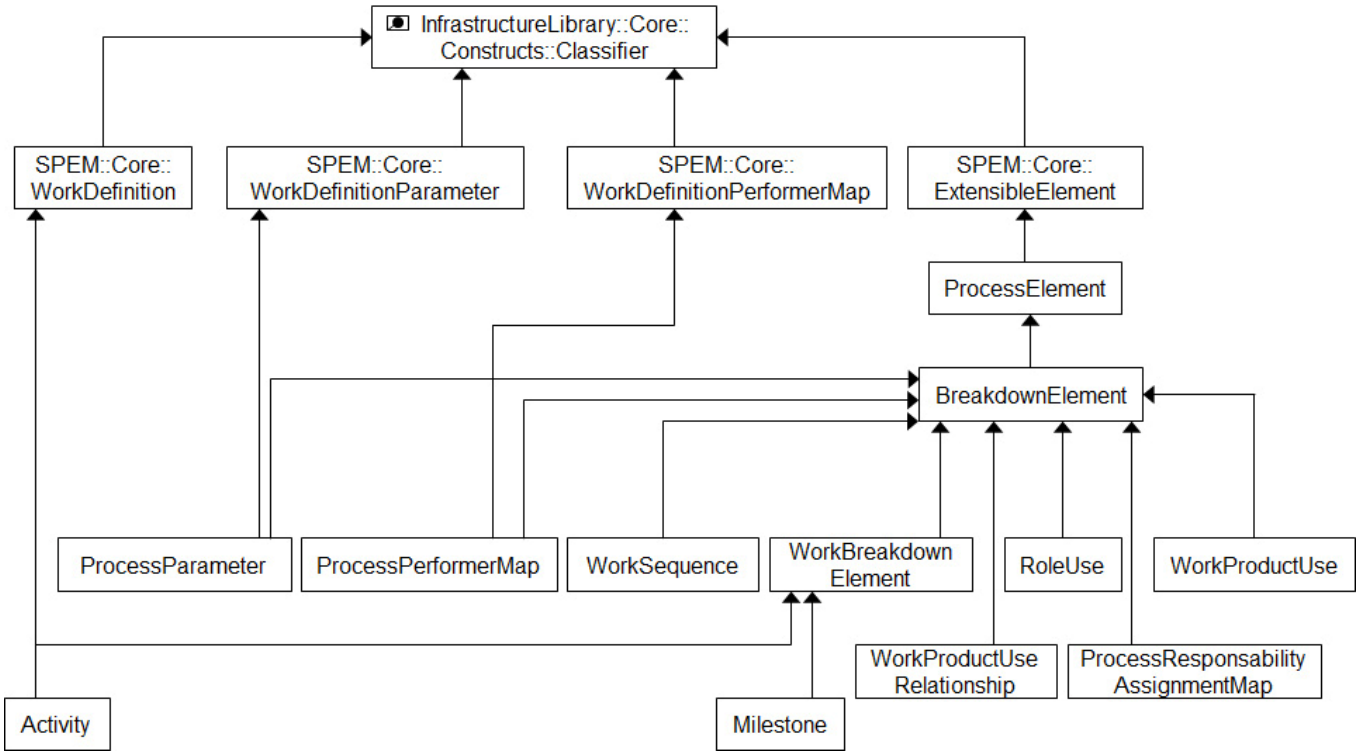


Fig. 8. Taxonomía del paquete *Process Structure* [13].

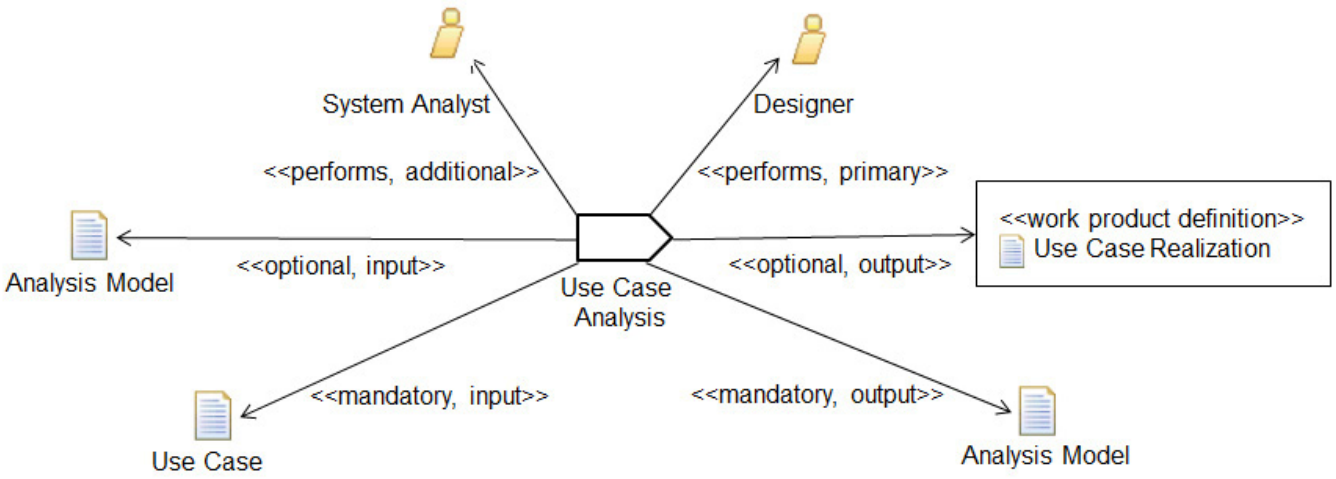


Fig. 9. Diagrama de clases UML 2 usando el perfil SPEM 2.0 [13].

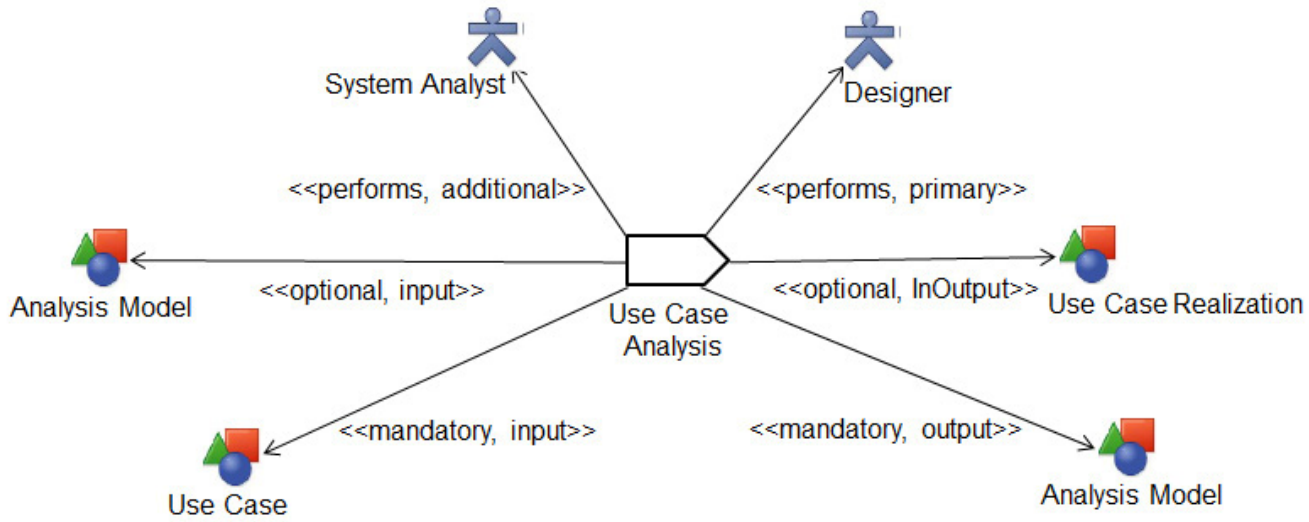


Fig. 10. El mismo diagrama de clases pero en SPEM 1.1 [13].

Es importante indicar que la Figura 10 no resulta válida para SPEM 1.1, debido a que las definiciones de trabajo (WorkDefinition) tienen que ser modeladas como operaciones de los roles. En cambio, en SPEM 2.0, las tareas y actividades son representadas con el estereotipo Actividad.

Para clarificar más la definición de procesos, SPEM 2.0 utiliza diagramas de clases, de actividades y de estado (Figura 11).

TABLA II. COMPARATIVA ENTRE LOS PERFILES SPEM 1.1 Y SPEM 2.0

Estereotipo	SPEM 1.1	SPEM 2.0
Activity		
Category		
Guidance		
ProcessComponent		
Rol		
Step		

Como resultado de las nuevas capacidades para la creación y gestión de procesos, las principales partes de la especificación están implementadas, por ejemplo en Eclipse Process Framework. Numerosas organizaciones y compañías han anunciado que modelarán sus procesos con esta tecnología (Armstrong Process Group, Open Group, Number Six Software, y Telelogic). Y otras compañías (como IBM, DSDM Consortium) han desarrollado productos comerciales.

III. LECCIONES APRENDIDAS

OMG propone el Metamodelo de Procesos de Ingeniería del Software (Software Process Enginnering Metamodel SPEM) para describir un proceso concreto de desarrollo de software o una familia de procesos de desarrollo de software relacionados. El propósito de SPEM es convertirse en un lenguaje estándar para el modelado de procesos de software.

SPEM permite modelar los procesos de desarrollo de software de una organización permitiendo así una comunicación efectiva y la aceptación por todo el equipo de desarrollo. Es un punto de referencia a partir del cual se puede mejorar, gestionar y automatizar todo el proceso de desarrollo. Los resultados de su utilización son procesos bien definidos que ayudan a mejorar la calidad, disminuir los costos y cumplir las metas.

SPEM está estructurado como un perfil UML y emplea muchos de sus diagramas (paquetes, casos de uso, clases, actividades, secuencias, diagramas de estado), excluye algunos elementos (nodos, componentes, etcétera) y agregan nuevos estereotipos (íconos).

El estándar SPEM está muy relacionado con UML, debido a que ambos están basados en la especificación MOF. SPEM se ubica en el metamodelo que está en el nivel M2 y sirve como plantilla para el nivel M1 de las cuatro capas arquitectónicas de abstracción definidas por la OMG.

El estándar SPEM 1.1 está construido dentro del paquete base de SPEM, el cual es un subconjunto de UML 1.4. Los elementos básicos, relaciones, y estructuras son definidos en el paquete base. Todos los otros elementos SPEM son definidos en relación a los elementos del paquete base a través de herencia. Debido a esta situación, SPEM es definido como un perfil (profile) UML en la documentación de SPEM.

El uso de SPEM permite una descripción abstracta de los elementos fundamentales del proceso de software (artefactos, roles, actividades, etc.) y la descripción de cómo ellos están relacionados con otros, facilitando su uso por el equipo de desarrollo. A nivel conceptual, la principal idea de la representación de procesos basados en SPEM es la relación entre tres elementos básicos: roles de procesos que son responsables y ejecutan actividades que consumen y producen productos de trabajo.

El modelado de procesos puede ser utilizado a diferentes escalas, la más simple: un proceso de software existente en una compañía debería ser modelado para una representación y entendimiento más claro. En el otro extremo, una gran compañía puede desear controlar una biblioteca de contenidos de procesos de software además que nuevos procesos pueden ser creados combinando y personalizando procesos componentes, ya sea desarrollados internamente o por terceros.

Se pretende que SPEM permita el intercambio entre herramientas UML y repositorios basados en MOF para el desarrollo de aplicaciones. Existen trabajos relacionados a esta temática [19] [2].También existen herramientas para el modelado de procesos de trabajo, lo que permite representar todos los elementos del proceso y favorecer la comunicación de los involucrados. Esto trae como consecuencia un modelado sencillo y fácil de modificar.

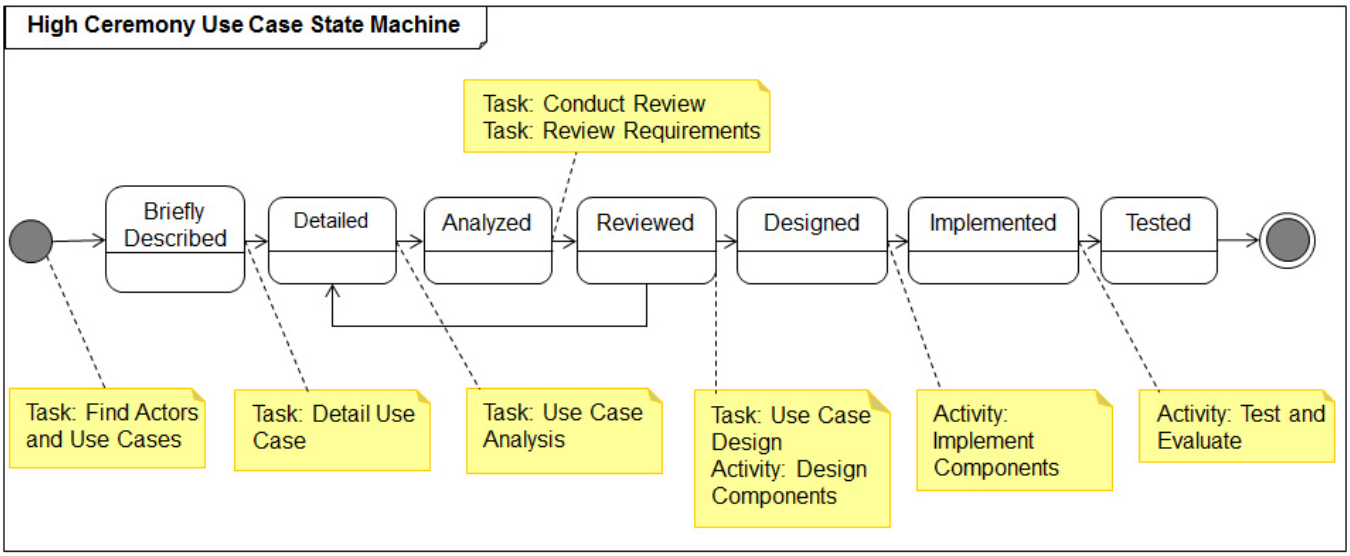


Fig. 11. Diagrama de estados [13].

SPEM ofrece numerosas ventajas: como por ejemplo ser independiente de la metodología o tecnología, estar basado en UML, además de favorecer la comunicación y el entendimiento de las partes involucradas en el proceso. Sin embargo también presenta algunas desventajas: como por ejemplo una semántica ambigua por estar basada en lenguaje natural.

La versión 1.1 ofrece una sintaxis rigurosa pero formaliza parcialmente la semántica, lo que origina algunas ambigüedades y no ofrece ayuda sobre cómo construir un modelo de proceso. El nuevo SPEM basado en MOF 2.0 proporciona una semántica más precisa.

IV. Conclusiones

SPEM es un estándar para el modelado de procesos de software propuesto por OMG en 2002. Es un lenguaje basado en UML que es independiente de la metodología empleada. Su objetivo es hacer un estándar para el modelado de procesos de desarrollo de software que unifique a la industria.

El uso de SPEM permite una descripción abstracta de los elementos fundamentales del proceso de software (artefactos, roles, actividades, etc.) y la descripción de cómo ellos están relacionados con otros, facilitando su uso por el equipo de desarrollo [16].

En general, sus principales características son [14] [16] [1] [7] [5]:

- Independiente de la metodología. SPEM es un metamodelo para definir procesos, desde los cuales otros procesos pueden ser construidos. La elección de la metodología de desarrollo que influirá el tipo de proceso es independiente de SPEM.
- Basado en UML. SPEM usa una orientación basada en objetos y la notación UML que es familiar a muchos profesionales del desarrollo de software.
- Independiente de la tecnología. SPEM proporciona una abstracción completa del proceso desde la infraestructura de implementación. Esto significa que puede modelar los procesos que mejor soporten la metodología seleccionada, sin preocuparse en la tecnología de implementación.
- Orienta en la automatización de procesos. Los beneficios de automatizar los procesos de desarrollo de software pueden ser obtenidos si estos procesos son modelados apropiadamente. SPEM está basado en estándares abiertos lo que permite usar una solución automatizada de procesos de cualquier vendedor que soporte el estándar.
- Incrementa la aceptación del desarrollador. La aceptación del equipo es clave para una efectiva implementación y manejo de cualquier proceso. SPEM ofrece una vista comprensiva y documentada de los procesos, de forma que los miembros de un equipo de desarrollo puedan entender y utilizar.

Se han hecho trabajos para automatizar flujos de trabajo en WFMS (WorkFlow Management System) utilizando SPEM [19]: Los procesos de software son modelados utilizando SPEM, entonces el metamodelo es transformado a la forma de XPDL (la interfaz estándar para la definición de procesos que separa la definición de un flujo de trabajo de su ejecución) por un motor de transformación, finalmente el modelo XPDL resultante es entregado a un motor XPDL como Shark.

Usar SPEM no es fácil porque la especificación es muy general y no proporciona directivas sobre cómo usarla. Además, su semántica está expresada en lenguaje natural, lo que origina la construcción de modelos de procesos que a veces son inconsistentes debido a la carencia de una definición formal de los conceptos. Se ha definido una especialización de SPEM que utiliza OCL para definir los conceptos claramente y expresar su semántica formalmente, sea a nivel metamodelo o a nivel de proceso [2].

La especificación SPEM versión 1.1 publicada en enero de 2005 no ha tenido una gran aceptación posiblemente a la falta de información para ponerlo en práctica, además que sólo es compatible con MOF 1.4 (la última versión es la 2.0). Otro punto en contra es que el metamodelo carece de una semántica precisa. La versión 2.0 está en proceso de revisión y se espera que resuelva esta situación y tenga una gran influencia práctica [10] [15].

Referencias

[1] M. Alanen, J. Lilius, I. Porres y D. Truscan, “Realizing a Model Driven Engineering process”, Techreport, Turku Centre for Computer Science, 2003. Recuperado de http://www.tucs.fi/publications/attachment.php?fname=TR565.pdf

[2] B. Combemale, A. Caplain, X. Crégut y B. Coulette, “Towards a rigorous use of SPEM”, ICEIS'06, Paphos(Cyprus), INSTICC, 2006. Recuperado de http://www.combemale.net/research/phd/2006/iceis250406-CCCC-poster401.pdf

[3] B. Combemale, S. Rougemaille, X. Crégut, F. Migeon, M. Pantel, C. Maurel y B. Coulette. “Towards a rigorous metamodeling”, MDEIS'06, Paphos (Cyprus), INSTICC, 2006. Recuperado de http://www.combemale.net/research/phd/2006/mdeis230406-CRCMPMC.pdf

[4] N. Debnath, D. Riesco, M. P. Cota, J. B. Garcia Perez-Schofield y D. R. M. Uva, D.R.M. “Supporting the SPEM with a UML extended workflow metamodel, Computer Systems and Applications, IEEE, 2006, pp. 1151 – 1154. Recuperado de http://ieeexplore.ieee.org/iel5/10748/33913/01618499.pdf

[5] E. Domiczi, “Customizing UML for software development process”, Nordic Workshop on UML based Software Development, Ronneby (Sweden), 2003. Recuperado de http://www.ipd.bth.se/consistencyUML/NordicWorkshop/papers/Customizing%20UML%20for%20Software%20Development%20Process.pdf

[6] M.-P. Gleizes, T. Millan y G. Picard, “ADELFE: using SPEM notation to unify agent engineering process and methodology”, Rapport interne IRIT no IRIT/2003-10-R, 2003. Recuperado de http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/docs/rapport2003-10-R.pdf

[7] C. Gonzalez-Perez, T. Mcbride y B. A Henderson-Sellers, “Metamodel for assessable software development methodologies”, Software Quality Journal, vol. 13, 2005, pp. 195–214, Springer Science + Business Media. Recuperado de http://www.springerlink.com/content/qm2u82716304v055/fulltext.pdf

[8] B. Henderson-Sellers y C. A. González-Pérez, “A comparison of four process metamodels and the creation of a new generic standard, Information and Software Technology, 47, 2005, pp. 49–65. Recuperado de http://linkinghub.elsevier.com/retrieve/pii/S0950584904000850.

[9] J. Hurtado y C. Bastarrica, “Hacia una Línea de Procesos Ágiles Agile SPsL”, Universidad del Cauca, 2005. Recuperado de http://www.dcc.uchile.cl/~cecilia/papers/AgileSPsL.pdf

[10] A. Järvi y T. Mäkilä, “Observations on modeling software processes with SPEM process components”, Proceedings of The 9th Symposium on Programming Languages and Software Tools, 2005. Recuperado de http://www.tucs.fi/publications/attachment.php?fname=inpJaMa05a.pdf

[11] T. Mäkilä, A. Järvi, “Spemmet – A tool for modelling software processes with SPEM”, Proceedings of the 9th International Conference on Information Systems Implementation and Modelling (ISIM '06), Přerov (Czech Republic), 2006. Recuperado de http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-180/paper09.pdf

[12] Object Management Group (OMG), “Software Process Engineering Metamodel (SPEM) Specification”, version 1.0 (formal/2002-11-14), 2002. Recuperado de http://www.omg.org

[13] Object Management Group (OMG), "Software Process Engineering Metamodel (SPEM) Specification", version 2.0 RFP (ad/2004-11-04), 2004. Recuperado de http://www.omg.org

[14] OSELLUS, "SPEM A standards-based approach for modeling software development processes", 2005. Recuperado de http://www.osellus.com/erms/resourceDetail.php?action=resource&rid=12

[15] J. M. Ribó y X. Franch, "A precedence-based approach for proactive control in software process modeling", Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering (SEKE '02), Ischia (Italy), ACM Press, 27, pp. 457-464, 2002. Recuperado de http://doi.acm.org/10.1145/568760.568839

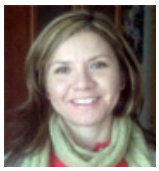
[16] A. Sampaio, A. Vasconcelos y P. Falcone, "Towards reconciling quality and agility in Web application development", International Workshop on Web Quality (WQ'04), International Conference on Web Engineering (ICWE04), Munich (Germany), 2004. Recuperado de http://www.elet.polimi.it/conferences/wq04/final/paper02.pdf

[17] W. Shengjun, J. Longfei y J. Chengzhi, "Represent Software Process Engineering Metamodel in Description Logic", Proceedings of World Academy of Science, Engineering and Technology, vol. 11, 2006, pp.109-113. Recuperado de http://staff.icar.cnr.it/staff/ruffolo/public_html/progetti/projects/23.Semantic%20BPM-%20in%20OntoDLP/Represent%20Software%20Process%20Engineering%20Metamodel%20in%20DL--v11-20.pdf

[18] Y. Wautelet, M. Kolp y Y. Achbany, "S-Tropos: An iterative SPEM-Centric software project management process", 2006. Recuperado de http://www.isys.ucl.ac.be/staff/youssef/Articles/WP_SPEM.pdf

[19] F. Yuan, M. Li y Z. Wan, "SPEM2XPDL: Towards SPEM Model Enactment". Software Engineering Research and Practice, 2006, pp. 240-245. Recuperado de http://ww1.ucmss.com/books/LFS/CSREA2006/SER5202.pdf

Víctor Hugo Menéndez Domínguez es Doctor en Tecnologías Informáticas Avanzadas por la Universidad de Castilla-La Mancha (UCLM), España; tiene un Máster en Tecnologías Informáticas por la misma institución. Además, cuenta con una Especialización en Docencia y una Licenciatura en Ciencias de la Computación por parte de la Universidad Autónoma de Yucatán (UADY), México. Es profesor titular en la Facultad de Matemáticas de la UADY. Su trabajo de investigación se centra en temas relacionados con la representación del conocimiento y el aprendizaje, así como la gestión de Objetos de Aprendizaje y Repositorios.



María Enriqueta Castellanos Bolaños es Maestra en Gestión de Tecnología de Información por la Universidad Anáhuac Mayab, México, con líneas de especialidad en Ingeniería de Software, Redes de Computadoras y Seguridad de Sistemas. Además, cuenta con una Especialización en Docencia y una Licenciatura en Ciencias de la Computación por parte de la Universidad Autónoma de Yucatán (UADY), México. Es profesora de tiempo completo en la Facultad de Matemáticas de la UADY. Su trabajo de investigación se centra en temas relacionados con la gestión del conocimiento y la Ingeniería Web.