Propuesta de un proceso de Enseñanza-Aprendizaje para la asignatura Diseño de Software, como proceso de software

María Inés Lund[1], Laura Nidia Aballay[1], María Claudia Gómez[2], Emilio Gustavo Ormeño[1]

[1] Universidad Nacional de San Juan, Instituto de Informática, San Juan, Argentina
[2] Universidad Nacional de San Juan, Departamento de Informática, San Juan, Argentina

Resumen

La cátedra Diseño de Software se dicta actualmente en 4º año de las carreras del Departamento de Informática de la Facultad de Ciencias Exactas, Físicas y Naturales (FCEFN) de la Universidad Nacional de San Juan (UNSJ). Esta materia se enfoca principalmente al Diseño Orientado a Objetos (DOO), brindando conceptos y conocimientos desarrollados en forma teórica y con un fuerte componente práctico, de todos los diagramas de modelado de software que provee el Lenguaje de Modelado Unificado (UML), con el fin de comprender acabadamente el objetivo que se persigue con cada uno de ellos y en qué casos es conveniente o útil aplicarlos.

El presente trabajo se sustenta de la experiencia adquirida en la práctica aplicada para la enseñanza de DOO, utilizando UML para el modelado, donde las actividades prácticas abarcan desde el análisis hasta llegar a una propuesta de diseño de implementación. Se presenta un modelo de proceso de enseñanza aprendizaje, como proceso de software, y los artefactos utilizados para guiar al alumno en la resolución de un problema de desarrollo de software específico, utilizando para su especificación el lenguaje de metamodelado de procesos SPEM 2.0 y para generar el modelado del proceso de software la herramienta Eclipse Process Framework Composer (EPFC).

Palabras clave: diseño orientado a objetos, modelo de procesos, procesos de software, proceso de enseñanza aprendizaje, UML, SPEM 2.0, EPF Composer

Abstract

The Software Design Chair is currently being taught in 4th year of the study programs offered by the Computing Science Department in the School of Hard, Physical and Natural Sciences (FCEFyN) of the National University of San Juan (UNSJ). This course mainly focuses on Object Oriented Design (OOD). It offers a theoretical development as well as a practical approach of the concepts and principles for all the software modeling diagrams provided by the Unified Modeling Language (UML). It aims at thoroughly understanding the objectives pursued by each model and in which cases they are more suitable or useful to be applied.

The present paper is based on the experience gained through the practical activities applied to the teaching of OOD by using UML for Modeling as well as learning tasks ranging from its analysis to a proposal for an implementation design. In addition a model for the teaching-learning process is presented, as software process, with all the artifacts used to steer the student in the resolution of a specific software development problem. The language used for specification is the process meta-modeling language SPEM 2.0 and the tool to generate the software process modeling is Eclipse Process Framework Composer (EPFC).

Keywords: object oriented design, process model, software process, teaching-learning process, UML, SPEM 2.0, EPF Composer.

1. Introducción

La cátedra Diseño de Software (en los planes de estudio anteriores denominada Sistemas de Información II y en los planes de estudios vigentes acreditados por CONEAU se denomina Ingeniería de Software II) se dicta actualmente en 4º año de las carreras Licenciatura en Ciencias de la Computación (LCC) y Licenciatura en Sistemas de Información (LSI) del Departamento de Informática de la Facultad de Ciencias Exactas, Físicas y Naturales (FCEFN). Los alumnos, a esta altura del cursado, ya tienen conocimientos de programación orientada a objetos y nociones elementales del paradigma, de modelado ágil y conceptos avanzados en ingeniería de requisitos.

Esta cátedra se enfoca principalmente al Diseño Orientado a Objetos (DOO) [1], [2] brindando

conceptos y conocimientos más desarrollados, en forma teórica y con aplicaciones prácticas, de todos los diagramas de modelado de software que provee UML [3], con el fin de comprender acabadamente el objetivo que se persigue con cada uno de ellos y en qué casos es conveniente o útil aplicarlos. La cátedra también abarca el Proceso Unificado de software (RUP) [4], Patrones de diseño [5] e Interacción humano computadora (HCI) [6]. Nociones de Arquitectura de software y diseño basado en componentes [7].

La actividad práctica, foco de nuestro estudio, consiste en la aplicación del proceso RUP, guiado por Casos de Uso (CU) [8], [9], a una narrativa de una situación problemática, cuya solución debe ser analizada y diseñada. Esta realidad que se plantea es acotada, pero con las características suficientes para que permita utilizar las distintas herramientas que provee UML para el modelado, siempre bajo el paradigma de la orientación a objetos.

Los alumnos se centran en una única realidad a modelar, por lo tanto, a medida que adquieren conocimiento de la misma y del dominio de aplicación, van obteniendo, a través de sucesivas iteraciones, y en forma incremental, diagramas más detallados y completos.

El proceso de enseñanza-aprendizaje presentado responde a las etapas del cursado de la asignatura, modelando, fundamentalmente, las actividades prácticas desarrolladas hasta llegar a una propuesta de diseño o modelo de implementación (sin la codificación) [7], utilizando un lenguaje de metamodelado de procesos.

El presente artículo está organizado de la siguiente manera, en la siguiente sección se muestra el Metamodelado del Proceso, donde se aborda SPEM 2.0 y EPF Composer. Posteriormente en la sección 3 se realiza una adaptación del proceso de enseñanza-aprendizaje en estudio al estándar SPEM, con el generador de modelos de procesos EPFC. Luego en la sección 4 se presenta el proceso de enseñanza-aprendizaje propuesto, como un proceso de software. Por último se presentan las conclusiones del trabajo.

2. Metamodelado del proceso

Un proceso es una entidad implícitamente o explícitamente definida para representar, adaptar, instanciar y ejecutar un conjunto de tareas parcialmente ordenadas, incluyendo sus participantes, entradas, salidas y conocimiento asociados, en un contexto de desarrollo o mantenimiento de software [10], [11].

Un modelo de proceso software es una abstracción o representación de un proceso, sea textual, gráfica o formal, en la que se capturan los aspectos más relevantes del mismo. Es descripto mediante un lenguaje de modelado y un lenguaje de modelado se basa en un metamodelo y una sintaxis concreta. Al igual que el software, los modelos de proceso son entidades complejas que requieren de varias vistas para mostrar ciertos aspectos del proceso. Por ejemplo: Actividades, Productos, Roles, Recursos [10].

2.1. SPEM 2.0 Software & Systems Process Engineering Metamodel specification (SPEM)

SPEM 2.0 es un estándar de metamodelado, que permite definir procesos de desarrollo de software, sistemas y sus componentes y sirve para representar procesos de ingeniería de software.

Su alcance se limita a los elementos mínimos necesarios para definir dichos procesos, sin añadir características específicas de un dominio o disciplina particular; No es un lenguaje de modelado de procesos en general, ya que está orientado a los procesos software. No provee conceptos propios para modelado del comportamiento, pero incluye mecanismos para enlazar el método externo elegido (diagramas de actividad de UML 2, BPMN/BPDM, etc).

Además de ser un metamodelo, SPEM es un marco de trabajo conceptual que provee los conceptos necesarios para modelar, documentar, presentar, publicar, gestionar, intercambiar y realizar métodos y procesos de software [12].

Para representar procesos se basa en tres elementos básicos: rol, producto de trabajo y tarea**.** Las tareas representan el esfuerzo a realizar, los roles representan quién lo realiza y los productos de trabajo representan las entradas que se utilizan en las tareas y las salidas que se producen. Básicamente un modelo de proceso consiste en decir quién (rol) realiza qué (tarea) para, a partir de entradas (productos de trabajo), obtener salidas (productos de trabajo) [13].

Tareas

Una Tarea describe una unidad de trabajo asignable y gestionable. Es la unidad atómica de trabajo para definir procesos. Su granularidad es de unas pocas horas a unos pocos días, afectando ciertos productos de trabajo y vinculando algunos roles. Define el trabajo realizado por roles.

Roles

Un Rol define un conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo. No se deben confundir roles con personas.

Productos de Trabajo

Un Producto de Trabajo es consumido, producido o modificado por Tareas. Existen tres tipos predefinidos de Productos de Trabajo:

Artefacto: de naturaleza tangible (modelo, documento, código, archivos, etc.). Un artefacto puede estar formado por otros artefactos más simples.

Entregable: 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

Resultado: un producto de trabajo de naturaleza intangible (resultado o estado), o que no está formalmente definido.

En la figura 1 se muestra la idea básica de SPEM, donde un rol es responsable de productos de trabajo que pueden ser la entrada o salida de una o más tareas.

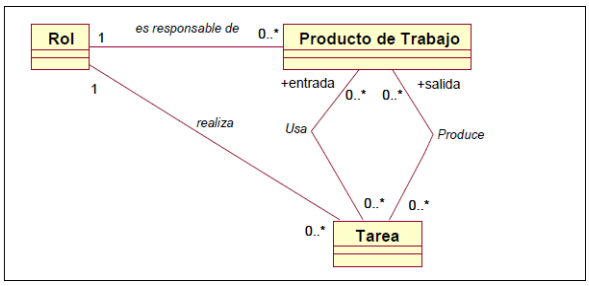

Figura 1. Idea básica de proceso en SPEM 2 [13].

2.2. Eclipse Process Framework Composer (EPF)

Esta herramienta se usó para generar un proceso de enseñanza-aprendizaje como un modelo de proceso de software, aplicado a las actividades prácticas de la asignatura diseño de software.

EPF Composer es un editor de procesos SPEM 2, permite generar marcos de procesos de software, para una organización o para obtener software de calidad. Es decir, tiene como objetivo producir un marco de procesos de software personalizable, soportando una amplia variedad de tipos de proyectos y estilos de desarrollo.

Es una herramienta gratuita, desarrollada dentro del entorno ECLIPSE, y permite generar automáticamente la documentación adecuada en formato para la web [14].

EPF tiene dos principales propósitos: uno, proveer a los practicantes de desarrollo una base de conocimiento del capital intelectual que les permita mirar, administrar y desplegar contenido y el segundo, proporcionar capacidades de ingeniería de proceso para proyectos de desarrollo concreto. EPF Composer ofrece catálogos de procesos predefinidos para las situaciones de un proyecto típico que pueden ser adaptadas a las necesidades individuales.

Roles, Tareas y Productos de Trabajo

EPF define métodos de desarrollo, como se muestra en la figura 2, mediante la identificación de roles de desarrollo que representan un conjunto de habilidades relacionadas, las competencias y responsabilidades de un equipo de desarrollo. Estos roles son responsables de determinados tipos de trabajo: por ejemplo, los desarrolladores son responsables de código fuente o analistas de sistemas son responsables de las especificaciones de casos de uso. Para crear y modificar productos de trabajo, se asignan tareas a los roles y estas tareas tienen como entrada y salida tipos específicos de productos de trabajo [15].

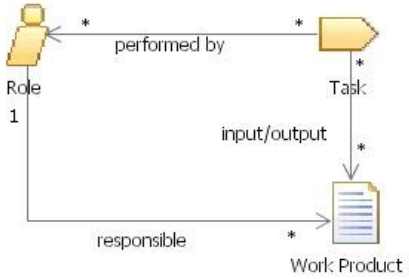

Figura 2. Modelo conceptual simplificado en EPFC [15].

Guías y Categorías

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

Las categorías permiten organizar e indexar la presentación del contenido: Si bien EPF ofrece categorías estándares, permite definir categorías propias para presentar la estructura y organización lógica de roles, tareas, productos de trabajo y guías [13], [15].

3. Adaptación del Caso de Estudio a EPF-SPEM

Para definir un proceso específico, desde un proceso base, se requiere adaptar las actividades y sus elementos relacionados. Es necesario agregar, quitar o modificar las actividades y salidas, teniendo en cuenta todos los insumos necesarios de un modelo de proceso base para desarrollar eficientemente un software de alta calidad. Los procesos específicos son un conjunto de actividades interrelacionadas, concretas, que se deben ejecutar a lo largo de la línea de tiempo del proyecto, tomando en consideración las características del mismo [16].

A continuación presentamos los elementos obtenidos a partir de adaptar las actividades desarrolladas en la ejercitación práctica de la asignatura Diseño de

software, como la identificación de roles, acciones, productos de trabajo, etc., generados con EPF Composer, [13].

3.1. Roles

Los roles identificados para el proceso de enseñanza-aprendizaje de la asignatura diseño de software fueron:

Analista: este rol es personificado por los alumnos, quienes desempeñan las tareas iniciales del proceso de diseño del software, necesarias para la comprensión y el alcance del sistema a desarrollar.

Cliente: en este caso el cliente es representado por el profesor de las clases prácticas, quien tiene conocimientos del modelo de negocio y de las necesidades del sistema en particular.

Diseñador: este rol también es caracterizado por alumnos de la asignatura, cumpliendo las tareas específicas de diseño de software.

Estos roles, generados en la herramienta, son mostrados en la figura 3. Tanto el rol de analista como el de diseñador utilizan UML [3] para el desarrollo de sus modelos.

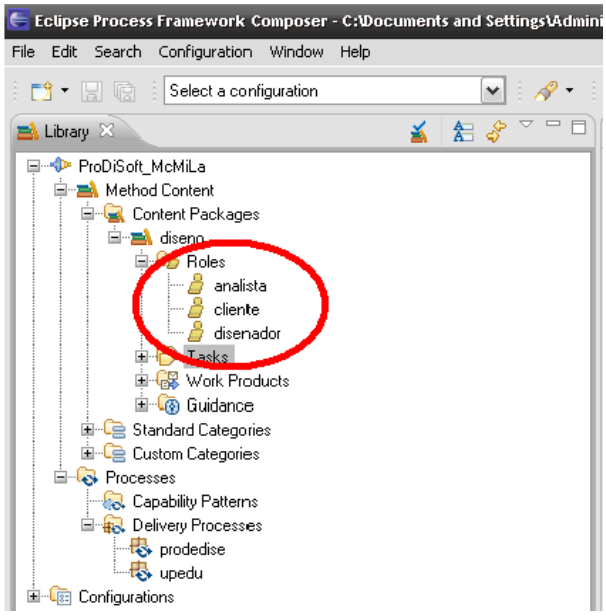

Figura 3. Roles del Proceso de Diseño de software

3.2. Tareas

Las tareas que se llevan a cabo en el proceso de análisis y diseño de software son:

- Comprensión del problema: se les entrega una narrativa para lectura y comprensión. El profesor evacua dudas, describe características y restricciones, acota el sistema a desarrollar.
- Generación del diagrama de casos de uso preliminar: a partir de la información relevada y el conocimiento adquirido.
- Elaboración del modelo conceptual: reconocimiento de los objetos y conceptos del dominio y sus relaciones.
- Refinamiento del modelo de casos de uso: en base al conocimiento adquirido y al análisis de las características del mismo.
- Documentación inicial de casos de uso: para especificar casos de uso utilizan la plantilla CUPIDo [17], [18].
- Elaboración del diagrama de clases preliminar: incorporación de atributos, multiplicidades, etc.
- Incorporación del flujo de eventos normal para cada caso de uso: en la plantilla CUPIDo.
- Incorporación del flujo de eventos alternativo para el manejo de excepciones (en la misma plantilla).
- Elaboración de diagrama de secuencias, en base a los flujos de eventos.
- Elaboración del diagrama de clases, en base al diagrama de clases preliminar y al modelo de casos de uso.
- Elaboración de diagramas de comunicación, siguiendo la línea de los d. secuencias.
- Elaboración del diagrama de estados, para aquellas clases que así lo ameriten.
- Elaboración del diagrama de tiempos, cuando la realidad lo justifique.
- Elaboración del diagrama de revisión de la interacción.
- Elaboración del diagrama de componentes, para analizar los componentes de software.
- Elaboración del diagrama de despliegue, donde se plantea la arquitectura de hardware que se plantea para la solución.

Estas tareas se pueden observar en la figura 4, generadas en la herramienta EPF.

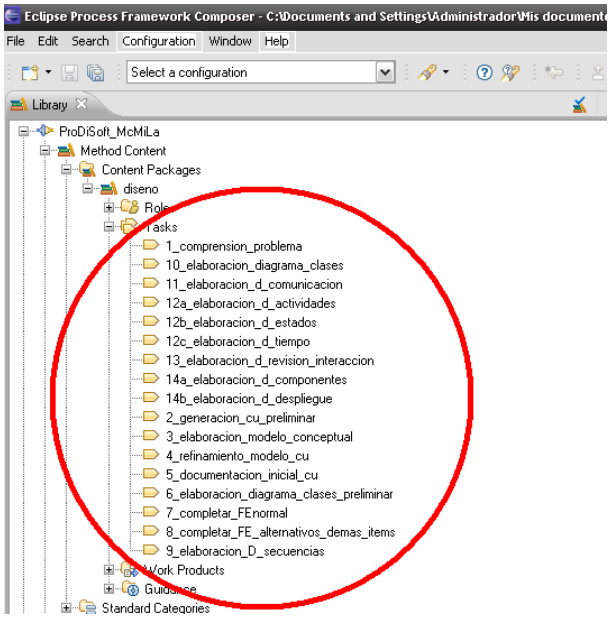

Figura 4. Tareas del Proceso de Análisis y Diseño

La ejecución de las tareas es iterativa, secuencial y algunas tareas podrían ejecutarse en paralelo y siempre con la posibilidad de volver atrás, iterar y refinar. Las mismas serán ejecutadas por diferentes roles y para realizarlas necesitan ciertos artefactos como entrada y tienen como resultado otros artefactos que pueden o no servir de entrada a otras tareas (productos de trabajo).

3.3. Productos de Trabajo

Tanto las entradas como las salidas de las tareas, como se expresó en el párrafo anterior, se materializan en forma de productos de trabajo, por ejemplo:

- Diagrama de casos de uso preliminar.
- Diagrama de casos de uso detallado.
- Diagrama de clases.
- Narrativa
- Plantillas de casos de uso, etc.

El detalle de los productos de trabajo identificados se puede observar en la figura 5.

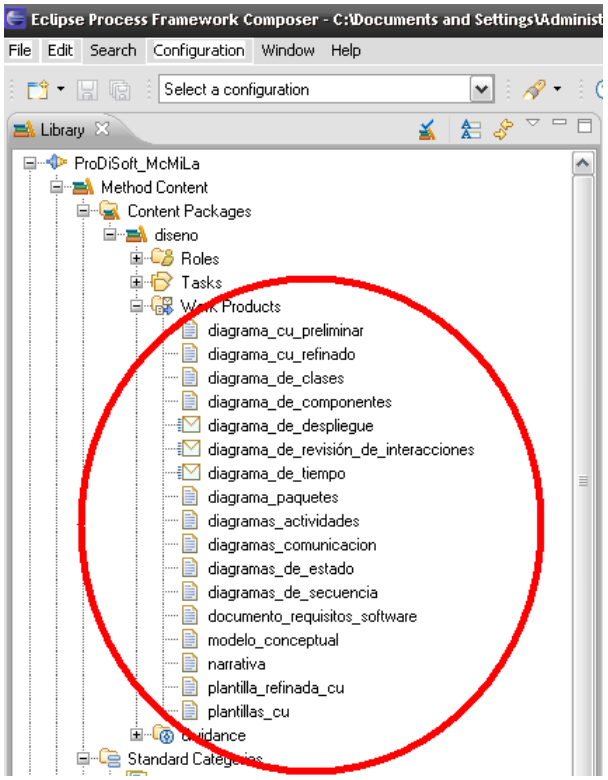

Figura 5. Productos de trabajo

3.4. Guías

Se identificaron todas las guías o instrumentos de apoyo necesarios para efectuar el proceso de análisis y diseño, dentro la asignatura Diseño de Software. Sólo a modo de ejemplo se presentan algunas de ellas en la figura 6.

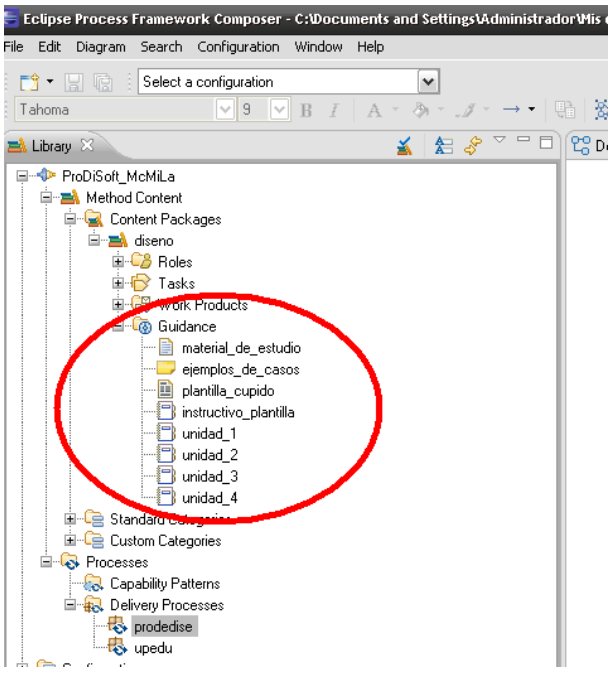

Figura 6. Guías

Entre ellas encontramos: las unidades teóricas de la asignatura, el modelo de plantillas CUPIDo, el instructivo de uso y aplicación de las plantillas, ejemplos de aplicación, etc.

3.5. Categorías

Se definieron categorías para agrupar o clasificar las tareas, en este caso de acuerdo al ciclo de vida de un software, bajo ciertas características:

- Análisis de requerimientos: es la etapa donde se realiza el relevamiento y elicitación de requisitos, participa el cliente.
- Diseño: es la etapa donde se conoce acabadamente la problemática, la solución y el sistema a desarrollar,
- Diseño de la implementación ó modelo de implementación: es la etapa donde se diseña el modelo de paquetes, de componentes y el de despliegue. No se llega a la codificación.

En la figura 7 se observa la personalización de categorías realizada en EPFC, con las tareas asignadas a cada una de ellas.

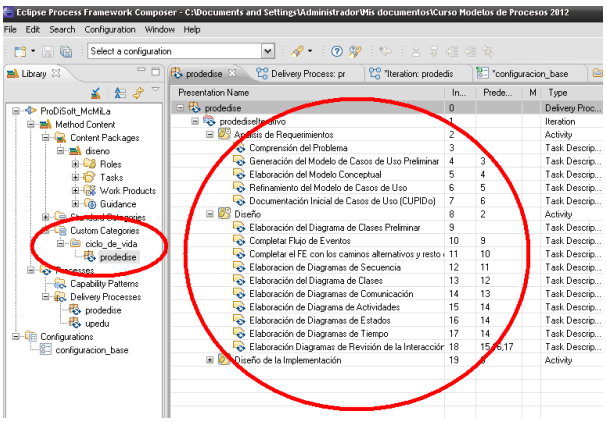

Figura 7. Categorías personalizadas

4. Proceso de enseñanza-aprendizaje propuesto

Se presenta en esta sección el proceso de enseñanza-aprendizaje, como si fuera un proceso de software, focalizado en la parte práctica de la asignatura Diseño de Software.

Este proceso será reflejado en varios modelos, cuyas figuras se presentan a continuación..

En la figura 8 se presentan las fases del proceso de enseñanza-aprendizaje, como si fuera un proceso de software, que se ejecutan en forma iterativa y se corresponden con las categorías, previamente definidas en la sección 3.5, del ciclo de vida de desarrollo del software [19].

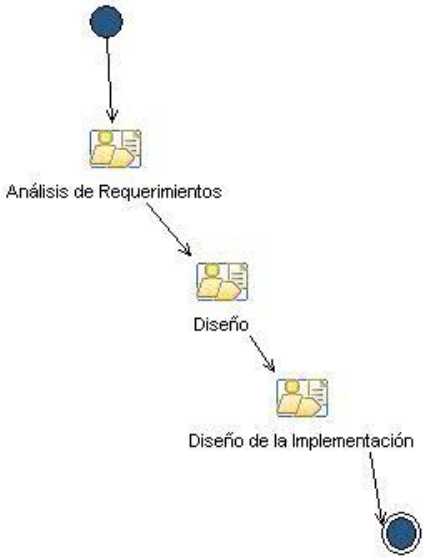

Figura 8. Modelo de Proceso de Software. Fases.

En cada una de estas fases se desarrollan tareas, estas fueron identificadas en la sección 3.2.

Para la fase Análisis de Requerimientos se asignaron las tareas específicas vinculadas a ella, como se muestra en la figura 9.

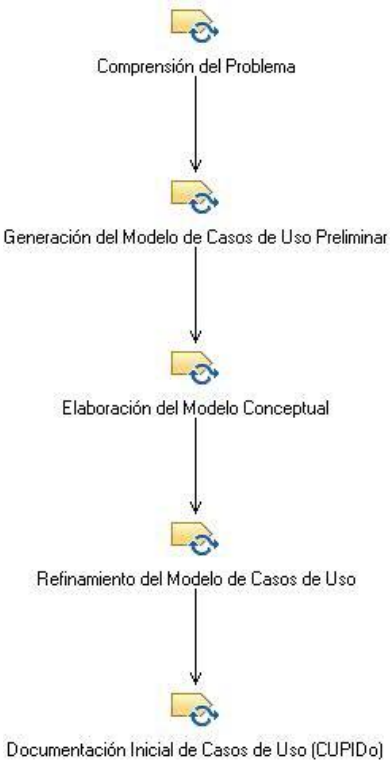

Figura 9. Tareas definidas para la fase Análisis de Requerimientos.

Cada una de esas tareas se realiza en forma iterativa e incremental, como la misma notación de EPFC para el símbolo lo indica.

En la figura 10 se muestran las tareas y cada uno de los productos de trabajo involucrados en la fase Análisis de Requerimientos, identificados en las secciones 3.2 y 3.3 respectivamente.

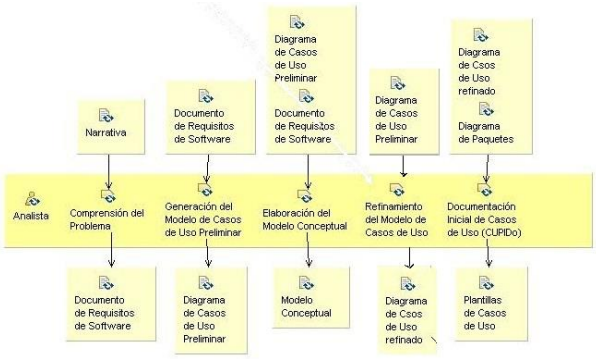

Figura 10. Tareas y productos de trabajo para el proceso de software de la fase Análisis de Requerimientos.

El rol de analista es el responsable de todas las tareas definidas en esta fase. Para la tarea *Comprensión del problema* se requiere como producto de trabajo de entrada la narrativa y produce como resultado el documento de requisitos de software. Este producto de trabajo es el insumo de la tarea siguiente *Generación del modelo de casos de uso preliminar*, cuyo producto de trabajo de salida es el diagrama de casos de uso preliminar. Siguiendo este criterio se pueden deducir los productos de trabajo insumidos y generados por cada una de las tareas definidas para la fase.

En la fase de diseño se incorporaron las tareas identificadas en la sección 3.2 referidas a esta categoría, las que se presentan en la Figura 11.

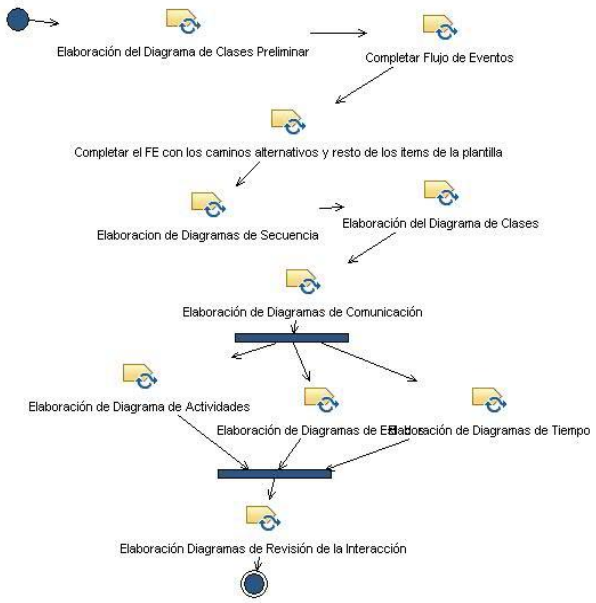

Figura 11. Tareas definidas para la fase Diseño.

Se puede observar que las tareas son de carácter iterativo e incremental y siguen un orden lógico secuencial. Se han usado los elementos notacionales de UML [3] para generar este diagrama, como si fuera un diagrama de actividades. Como ya se explicó anteriormente cada una de estas tareas genera productos de salida que son insumos de las tareas subsiguientes, y que se ejecutan en forma incremental. Después de la elaboración de los diagramas de comunicación pueden realizarse, en forma paralela o de forma indistinta (ya que no hay dependencia entre ellas), los diagramas de actividades, diagramas de estados y diagramas de tiempos [5]. Si, es necesario haber terminado con todos ellos para elaborar el diagrama de revisión de la interacción, que se nutre de todos los diagramas de comportamiento e de interacción generados previamente.

En la figura 12 se exponen las tareas de la fase Diseño de la Implementación identificadas en la sección 3.2. Esta fase, también conocida como modelo de implementación, es realizada al final del proceso de enseñanza-aprendizaje propuesto. Es necesario aclarar que esta fase podría realizarse previo a la fase de diseño, como un diagrama de arquitectura general del proyecto.

Figura 12. Tareas definidas para la fase Diseño de la Implementación.

Los alumnos, al finalizar todas las actividades desarrolladas en el marco del proceso de enseñanza-aprendizaje, como un proceso de software, no sólo logran aprender el uso y aplicación de cada modelo y diagrama de UML útiles para el análisis y diseño de un sistema, sino que llegan a la comprensión acabada del sistema a implementar. Esta metodología también les facilita la conceptualización de los contenidos teóricos

Conclusiones

Desde hace varios años se trabaja, en la asignatura Diseño de Software, con esta estrategia de ejecución de las actividades prácticas, en donde los alumnos realizan las etapas de análisis y diseño (llegando al modelo de implementación), usando UML para el modelado. Estas actividades se realizan para un solo sistema o narrativa por vez. Cuando se terminan con todas las tareas definidas en el proceso, recién se comienza con otra realidad para modelar y reiniciar el proceso descripto.

Para los autores la elaboración del presente trabajo nos ha llevado a analizar la ejercitación práctica identificando las tareas, las fases (categorías), los productos de trabajo insumidos y resultantes, especificando las guías de apoyo requeridas para realizar cada tarea, analizar la vinculación entre tareas, productos de trabajo y guías, ha permitido por un lado tomar conocimiento detallado del proceso de enseñanza-aprendizaje con el que se trabaja. Por otra parte, adaptar cada uno de ellos usando la especificación de metamodelado SPEM 2.0, a través de la framework EPC Composer ha posibilitado formalizar la estrategia de trabajo del proceso de enseñanza-aprendizaje de la cátedra, como si fuera un proceso de software.

Para los alumnos esta forma guiada de ejecución de las actividades prácticas, donde van adquiriendo conocimiento del sistema a desarrollar en forma incremental, desde lo más abstracto a lo más detallado, les permite entender o comprender los conceptos teóricos del diseño orientado a objetos, usando como lenguaje de modelado a UML, a través de la experiencia práctica de un diseño de software específico, logrando el razonamiento, a través de la vinculación del concepto teórico con la práctica ejecutada, sobre qué modelar, cuándo y cómo hacerlo.

El contar con herramientas y técnicas de modelado permitirá mejorar la dinámica de las actividades prácticas de la asignatura, incorporando tareas, productos de trabajo o guías, para facilitar al alumno en la conceptualización de contenidos a través de la ejercitación práctica.

Este proceso de enseñanza-aprendizaje para la asignatura diseño de software, como proceso de software, es una propuesta, queda pendiente la realización de un estudio empírico sobre cuanto mejora el aprendizaje de los alumnos siguiendo un proceso de enseñanza-aprendizaje como el planteado, o comparativo con otros procesos de enseñanza-aprendizaje existentes.

Agradecimientos

Este trabajo ha sido realizado por el equipo de cátedra de la asignatura “Diseño de Software” del Departamento de Informática y en el marco del proyecto E883/2010: “Formalización de descripciones de casos de uso a través de metamodelos – ForCUPIDo” del Instituto de Informática, ambos de la FCEFN de la UNSJ.

Referencias

[1] G. Booch, J. Rumbaugh, y I. Jacobson, *The Unified Modeling Language User Guide*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1999.

[2] I. Jacobson, *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Professional, 1992.

[3] «Object Management Group - UML». [En línea]. Disponible en: http://www.uml.org/. [Accedido: 17-nov-2010].

[4] I. Jacobson, G. Booch, y J. Rumbaugh, *The unified software development process*. Reading, Mass.: Addison-Wesley, 1999.

[5] C. Larman, *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. Upper Saddle River (New Jersey): Prentice-Hall, 2005.

[6] C. Faulkner, *The Essence of Human-Computer Interaction*. Pearson-Prentice Hall, 1998.

[7] H. Züllighoven, R. F. Beeger, y D. Bäumer, *Object-oriented construction handbook developing application-oriented software with the tools & materials approach*. Amsterdam; Boston; San Francisco; Heidelberg: Elsevier; Morgan Kaufmann; Dpunkt.verlag, 2005.

[8] I. Jacobson, G. Booch, y J. Rumbaugh, «Chapter 1: The Unified Process: Use-Case Driven, Architecutre-Centric, Iterative, and Incremental», en *The Unified Software Development Process*, Addison-Wesley Professional, 1999, p. 5.

[9] D. Kulak y E. Guiney, *Use Cases: Requirements in Context*, 2.ª ed. Addison-Wesley Professional, 2008.

[10] S. T. Acuña y X. Ferré, «Software Process Modelling», en *proc. World Multiconf. on Systemics, Cybernetics and Informatics*, Orlando, FL, 2001, pp. 237–242.

[11] J. A. Hurtado Alegría, M. C. Bastarrica, A. Quispe, y S. F. Ochoa, «An MDE approach to software process tailoring», en *Proceedings of the 2011*

International Conference on Software and Systems Process, New York, NY, USA, 2011, pp. 43–52.

[12] OMG, «Software & Systems Process Engineering Metamodel specification (SPEM) Version 2.0». abr-2008.

[13] Francisco Ruiz, Javier Verdugo, «Guía de Uso de SPEM 2 con EPF Composer». Universidad de Castilla La Mancha, 01-abr-2008.

[14] ECLIPSE, *Eclipse Process Framework Project (EPF)*. ECLIPSE, 2012.

[15] Haumer, Peter, «Eclipse Process Framework Composer». IBM, abr-2007.

[16] H. Washizaki, «Building Software Process Line Architectures from Bottom Up», en *Product-Focused Software Process Improvement*, J. Münch y M. Vierimaa, Eds. Springer Berlin Heidelberg, 2006, pp. 415-421.

[17] M. I. Lund, C. Ferrarini, L. Aballay, y E. Meni, «CUPIDo - Plantilla para Documentar Casos de Uso», presentado en V Congreso de Tecnología en Educación y Educación en Tecnología, El Calafate, Santa Cruz, Argentina, 2010.

[18] M. I. Lund, L. N. Aballay, E. Torres, M. Herrera, y E. G. Ormeño, «Validación de usabilidad de una plantilla para documentar casos de uso – estudio exploratorio», presentado en XVII Congreso Argentino de Ciencias de la Computación, 2011.

[19] A. Weitzenfeld, *Ingeniería de software orientada a objetos con UML, Java e Internet*. Cengage Learning Editores, 2005.

Dirección de Contacto del Autor/es:

María Inés Lund
Av. Ignacio de la Roza 590 (O)
Complejo Universitario "Islas Malvinas"- J5402DCS
Rivadavia, San Juan, Argentina
mlund@iinfo.unsj.edu.ar

Laura Nidia Aballay
Av. Ignacio de la Roza 590 (O)
Complejo Universitario "Islas Malvinas"- J5402DCS
Rivadavia, San Juan, Argentina
laballay@iinfo.unsj.edu.ar

María Claudia Gómez
Av. Ignacio de la Roza 590 (O)
Complejo Universitario "Islas Malvinas"- J5402DCS
Rivadavia, San Juan, Argentina
cacu_gomez@yahoo.com.ar

Emilio Gustavo Ormeño
Av. Ignacio de la Roza 590 (O)
Complejo Universitario "Islas Malvinas"- J5402DCS
Rivadavia, San Juan, Argentina
eormeno@iinfo.unsj.edu.ar

María Inés Lund. Lic en Informática. Esp en Sistemas de Información para Intrantets. Mg en Informática. Profesor Adjunto Exclusivo a cargo de la práctica de la asignatura Diseño de Software. Investigadora del Instituto de Informática

Laura Nidia Aballay. Lic en Ciencias de la Información. Actualmente realizando Maestría en Informática. Profesor Jefe de Trabajos Prácticos Exclusivo asistente de práctica de la asignatura Diseño de Software. Investigadora del Instituto de Informática

María Claudia Gómez. Lic en Administración de Empresas. Lic en Investigación Operativa. Mg en Logística Industrial. Profesor Titular Exclusivo responsable de la asignatura Diseño de Software. Investigadora del Departamento de Informática

Emilio Gustavo Ormeño. Lic en Informática. Esp en Sistemas de Información para Intrantets. Mg en Informática. Profesor Adjunto Exclusivo. Colaborador en la asignatura Diseño de Software. Investigador del Instituto de Informática