Nuevo Enfoque para la Enseñanza del Paradigma MDD: Ingeniería de Requerimientos Basada en Modelos apoyada por Tecnologías CASE

Leopoldo Nahuel1,2 , Cecilia Ariste1 , Roxana Giandini1,2
1 LINSI - Facultad Regional La Plata - Universidad Tecnológica Nacional, La Plata, Buenos Aires, Argentina
2 LIFIA - Facultad de Informática - Universidad Nacional de La Plata, La Plata, Buenos Aires, Argentina
{lnahuel, cariste, rgiandini}@frlp.utn.edu.ar

Resumen

El Desarrollo Dirigido por Modelos (MDD, Model Driven Development) es hoy un paradigma innovador y pujante de la Ingeniería de Software, su proceso radica en la construcción y transformación de modelos con distintos niveles de abstracción. Por su parte, las actividades y productos de la Ingeniería de Requerimientos son un eslabón fundamental para las distintas disciplinas de trabajo de la Ingeniería de Software. Más precisamente la Ingeniería de Requerimientos Basada en Modelos (IRBM), apunta a la conceptualización y especificación de los requerimientos a través de múltiples modelos gráficos y/o textuales. Tanto la IRBM como MDD, requieren soporte de tecnologías CASE (Computer Aided Software Engineering) para construir distintas vistas de modelos y establecer conexión entre ellos conocida como trazabilidad de modelos. Este trabajo expone los beneficios de enseñar IRBM como parte de la ejecución de las etapas iniciales del paradigma MDD, apuntando hacia identificación y conceptualización de requerimientos a través de múltiples modelos. Esta integración metodológica-tecnológica entre las bases de MDD con técnicas de IRBM y el apoyo de herramientas CASE, permite explotar de manera práctica el autoaprendizaje de conceptos teóricos comúnmente impartidos en asignaturas de Ingeniería de Software: escalabilidad y trazabilidad de modelos, evaluación de impactos a partir de cambios de requerimientos, refinamiento de modelos funcionales en modelos de diseño y estimaciones de proyectos a partir de modelos funcionales.
Palabras clave: Enseñanza de Ingeniería de Requerimientos, Autoevaluación del Aprendizaje, Desarrollo Dirigido por Modelos (MDD), Ingeniería de Requerimientos Basada en Modelos (IRBM), Herramientas CASE, Trazabilidad de Modelos.

Abstract

Currently, Model Driven Development (MDD) constitutes an innovative paradigm of the Software Engineering. Its process is based on the construction and transformation of models into different levels of abstraction. For its part, the activities and products of Requirement Engineering (RE) are considered today a fundamental link in several work disciplines of Software Engineering. More precisely Model Based Requirement Engineering (MBRE), points to conceptualization and specification of requirements across multiple graphic or textual models. IRBM and MDD require support of CASE technology (Computer Aided Software Engineering) to build varied model's view and establish connection between them; it's well-known as model traceability. This paper describes the benefits of teaching MBRE as part of the execution of the initial steps of the MDD paradigm, pointing to identification and conceptualization of requirements through numerous models. This methodology-technology integration between the bases of MDD, IRBM's techniques and CASE tools support, allow explode in a practical way the self-study of common theorical concepts given in Software Engineering courses : scalability and traceability of models; impact assessment of requirement changes; refining functional artifacts and project management in MDD.
Keywords: Requirements Engineering Education, Self Learning, Model Driven Development (MDD), Requirements Engineering Based Models (IRBM), CASE Tools, Traceability Model.

1. Introducción

En la actualidad existen problemáticas de las ciencias de la computación englobados en lo que se conoce "Crisis del Software", que se vienen acarreando de la década del 60'. Algunas de estas son: dificultades para la planificación del proceso, lo que deriva en tiempos de desarrollo extensos y encarecimiento del proyecto; problemas con el mantenimiento; compatibilidad y actualización; entre otros. Es así como surge la Ingeniería del Software, para dar solución a los principales problemas del proceso de desarrollo de software. Actualmente, esta disciplina cuenta con un cuerpo muy grande de conocimientos y herramientas que nos posibilitan afrontar la creación de grandes aplicaciones computacionales. El primer gran

cambio que se propone, dentro de esta rama de la ingeniería, es tomar al desarrollo de software de la misma manera que se aborda un proyecto ingenieril de cualquier otra especialidad. Con esta idea surgen varias técnicas de análisis y diseño de sistemas que dan soporte al relevamiento y el entendimiento del dominio, en conjunto con lenguajes de modelado estándar que permiten la esquematización y la documentación de sistemas complejos.

En este contexto y bajo la premisa de que los requerimientos completos, no ambiguos y rastreables, promueven la eficiente administración del cambio, evolución y mantenimiento del producto, surge la Ingeniería de Requerimientos.

Muchas técnicas y herramientas se han creado para acrecentar el cuerpo de conocimientos que esta integra. En particular, la Ingeniería de Requerimientos Basada en Modelos (IRBM), es una de las más importantes. Esta integra un conjunto de conocimientos que se aplican durante la primera etapa del desarrollo, apuntando a obtener un relevamiento del dominio y el problema en sí, lo más acertado posible, de manera de entender qué es lo que se requiere hacer. Esta etapa es indispensable si se pretende lograr una buena solución tecnológica acorde a las necesidades de los usuarios, y es por esto, que la IRBM toma fuerte relevancia en el campo de la ingeniería de software.

A su vez, somos testigos del surgimiento de un nuevo paradigma en la ingeniería de software, que se propone metas muchos más altas que las que el enfoque tradicional, basado en modelos, proponía. Hablamos de Desarrollo Dirigido por Modelos. (MDD por sus siglas en inglés). MDD propone que sean los modelos los que dirijan el desarrollo entero durante todas sus etapas. De esta manera partimos de los modelos con alto nivel de abstracción llegando a los más concretos, logrando finalmente la generación del código fuente, a través de transformaciones sucesivas de modelos.

Este trabajo propone un nuevo enfoque para la enseñanza del paradigma MDD a través de la puesta en práctica de las principales herramientas y métodos de la IRBM en un proyecto de modelado utilizando tecnologías CASE. Dejando así en claro la utilidad de este enfoque para dar los primeros pasos en la comprensión del nuevo paradigma dirigido por modelos, logrando especificar los primeros modelos (CIM - PIM) de un proceso MDD, estimulando así el aprendizaje con incorporación de tecnología.

1.1 Organización de este artículo

Este documento se organiza de la siguiente manera: en la sección 2 se introducen los principales conceptos y los lineamientos generales que dan contexto al trabajo. En la sección 3 se describe el marco de enseñanza de IRBM propuesto para la enseñanza superior. La sección 4 se presenta el desarrollo de un Caso de Estudio aplicando IRBM con Herramientas CASE mientras que en la sección 5 se mencionan los aportes de la IRBM en la enseñanza de MDD. En la sección 6, se describen los resultados y experiencia obtenidos hasta el momento. Finalmente, en la sección 7 presentan las conclusiones y líneas de trabajo futuro.

2. Marco Teórico

En esta sección introducimos los principales conceptos, paradigmas, estándares y herramientas que dan base al propósito del presente trabajo.

2.1. MDD: una metodología de trabajo para la producción de aplicaciones software

El Desarrollo Dirigido por Modelos MDD, surge como una idea innovadora para dar solución a problemas clásicos en el ámbito de la Ingeniería de Software.

El Paradigma MDD [1] brinda una nueva forma de construir sistemas de software, situando a los modelos como los artefactos principales del proceso de desarrollo; la idea principal subyace en la obtención automática de código fuente por medio de transformaciones automáticas aplicadas a los modelos completos y consistentes.

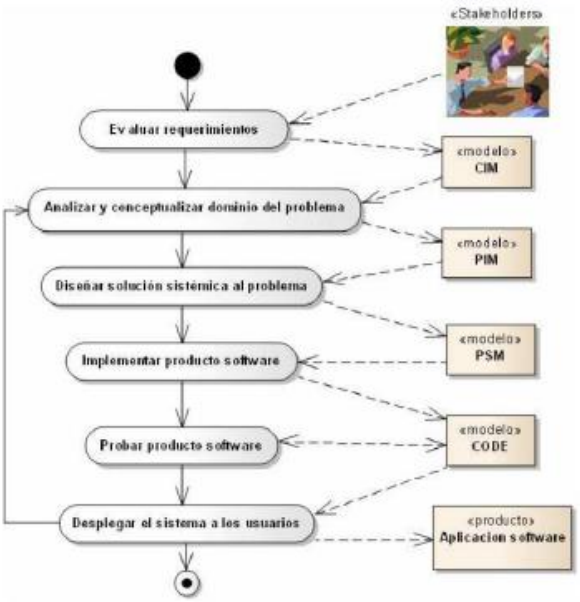

Figura 1: Ciclo de vida del software en MDD.

Estas transformaciones (CIM PIM PSM IM/CODE) nos llevan desde los niveles más abstractos hacia los más concretos, generando los modelos correspondientes para cada nivel de abstracción. En la Figura 1 se muestra en forma ilustrativa el proceso iterativo e incremental con los correspondientes modelos y sus transformaciones.

Cabe destacar que estas transformaciones son llevadas a cabo automáticamente por herramientas computacionales, librando de estas tareas al equipo de desarrollo.

MDD establece 4 tipos de modelos diferentes, como se puede observar en la Figura 2:
CIM (Computer Independent Model): Tienen un alto nivel de abstracción y son usados para describir la lógica del dominio del negocio, desde una visión independiente de la computación.
PIM (Platform Independent Model): estos modelos describen en forma abstracta la funcionalidad del sistema independientemente de la tecnología.
PSM (Platform Specific Model): modelos que especifican el sistema en términos de una plataforma más específica.
IM (Implementation Model) o CODE: especifica el sistema a través del código fuente en una tecnología específica.

Las transformaciones pueden especificarse por muchos lenguajes, inclusive los de propósito general. Sin embargo la tendencia de MDD es hacer uso de lenguajes dedicados a la transformación específica de modelos [2].

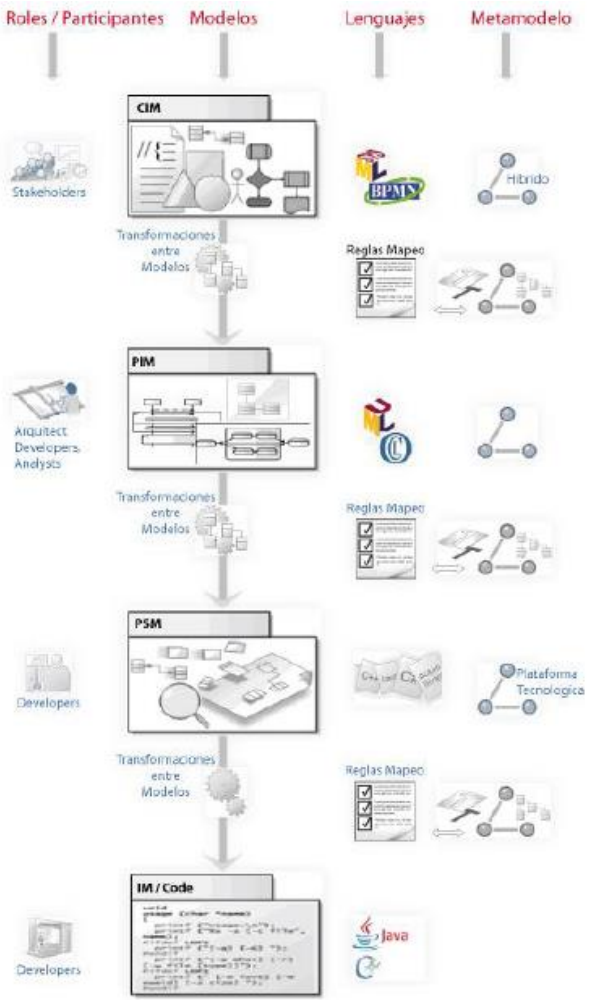

Figura 2: Evolución y transformación de los modelos en el ciclo de vida MDD.

Este paradigma ofrece a la Ingeniería de Software una alternativa, con un enfoque diferente, para la construcción de sistemas de software, cuyo factor principal es la manipulación del código fuente de los lenguajes de programación por sobre los modelos.

2.2. Lenguajes de modelado estándar

La escritura de los modelos se realiza desde etapas tempranas del proceso de desarrollo, partiendo de una especificación inicial del dominio y la Ingeniería de Requerimientos, pasando por diferentes niveles de abstracción hasta llegar al código fuente a través de las transformaciones. Para la escritura de los modelos es indispensable el aprendizaje de lenguajes de modelado que nos provean de una simbología sintácticamente robusta, y a la vez que tengan la suficiente precisión para poder realizar de forma correcta la transformación entre modelos. Por tal motivo se propone el uso de lenguajes ampliamente utilizados en la industria y estandarizados, como UML.

Dado que UML es un lenguaje de propósito general, está provisto de una gran variedad de símbolos, artefactos y diagramas que lo hacen flexible y muy abarcativo, independientemente de la complejidad o problemática del sistema que se modele. Pero es esta misma generalidad lo que hace que se pierda precisión sobre el significado semántico en algunos aspectos más finos del diseño. Esto motiva al uso de Lenguajes de Modelado Propósito Específico (LMPE) en complemento de los Lenguajes de Modelado de Propósito General (LMPG), para definir modelos precisos que permitan la correcta transformación de modelos destinados a la generación de productos finales [5]. Como parte de la propuesta se promueve el uso de los siguientes lenguajes:

UML (Unified Model Language): es un lenguaje de modelado para sistemas de software. Es un lenguaje gráfico que permite visualizar, especificar, construir y documentar sistemas de software. UML, permite a los desarrolladores visualizar los resultados en esquemas o diagramas estandarizados.

SysML (System Model Language): lenguaje de modelado de propósito general para sistemas de software y no software, es un lenguaje gráfico, con una notación estándar para el modelado dentro de un dominio específico de aplicaciones en el campo de la Ingeniería de Sistemas. Es un lenguaje gráfico, que reutiliza un subconjunto de UML y proporciona extensiones adicionales para satisfacer los requerimientos del lenguaje. Este lenguaje de Modelado de Sistemas soporta la especificación, análisis, diseño, verificación y validación de una amplia gama de sistemas y subsistemas. Este lenguaje agrega un tipo de diagrama, sumamente importante para la IRBM, los diagramas de Requerimientos [6, 7,8].

OCL (Object Constraint Language): lenguaje formal y textual (no gráfico) para especificar restricciones y reglas de negocio sobre los modelos orientados a objetos. Forma parte del estándar UML. Fue estandarizado por OMG y se lo definió como un lenguaje que posibilita a los desarrolladores a la escritura de expresiones semánticas del sistema, que no pueden ser expresadas por medio de UML. Los modelos de UML, son complementados con expresiones OCL, quitando ambigüedades y haciendo los modelos más precisos [9,10].

2.3. Proceso de desarrollo y herramientas en la IRBM desde un enfoque educativo

Existen distintas metodologías tradicionales para el desarrollo de software. Entre las principales podemos mencionar a RUP (Rational Unified Process) [11] y MSF (Microsoft Solution Framework), que centran su atención en llevar una documentación exhaustiva de todo el proyecto y cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo.

La metodología RUP, bien conocida en el ámbito académico de ciencias informáticas y computación, promueve el desarrollo iterativo, poniendo énfasis en los requerimientos, análisis y diseño de sistemas. Este se ha diseñado conjuntamente con UML, por lo que la descripción del flujo de trabajo se orienta alrededor de distintos artefactos del lenguaje UML.

Sin embargo, para este trabajo se consideró conveniente utilizar el proceso OpenUP [12] para ilustrar el desarrollo de un Caso de Estudio aplicando IRBM apoyado con tecnología CASE, que será detallado en la sección 4. La elección de OpenUP se basa en que es un proceso mínimo y suficiente de la concepción RUP, lo que significa que solo el contenido imprescindible y necesario es incluido como parte del esquema metodológico para llevar adelante un proyecto de desarrollo. Esto último, es un factor valioso al adecuarlo a contextos de Enseñanza de Ingeniería de Software e Ingeniería de Requerimientos, desde enfoques orientados a modelos. OpenUP es parte de Eclipse Process Framework (EPF) [25], un proceso de desarrollo open source dentro de Eclipse Foundation.

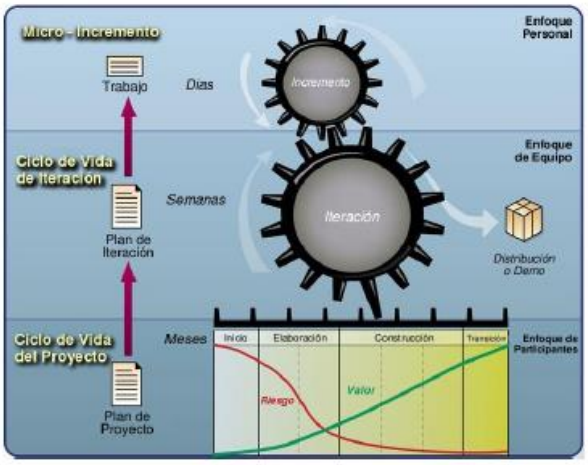

Figura 3: Las tres capas y workflow de trabajo del proceso OpenUP

En la Figura 3, puede verse el ciclo de vida del proceso OpenUP. Este no ofrece lineamientos para todos los artefactos que se manejan en un proyecto pero tiene los componentes básicos que sirven de base a procesos específicos, dependiendo del dominio del negocio. La mayoría de los artefactos de OpenUP están declarados para promover el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances. Además puede aplicarse en distintos proyectos de desarrollo de software.

Junto a estas metodologías, la industria informática ha desarrollado un soporte automatizado para la construcción y mantenimiento de software. Estas herramientas son las llamadas Computer Aided Software Engineering, más conocidas por sus siglas en inglés CASE.

En virtud de las incumbencias curriculares de la carrera Ingeniería en Sistemas de Información, que involucra el estudio de lenguajes de modelado, especificación de modelos precisos, herramientas para edición/transformación de modelos y lenguajes para especificar modelos formales, entre otros, yace la necesidad de usar una herramienta CASE en el ámbito de enseñanza académica. La edición visual de diagramas UML, SysML, BPMN permiten aprender y madurar el uso adecuado de lenguajes de modelado. Las herramientas CASE modernas y robustas, nos dan lugar a ejecutar acciones para el proceso de enseñanza aprendizaje de las reglas de buena formación en distintos aspectos del modelado de software, desde etapas tempranas del ciclo de vida de sistemas, permitiendo:
a) lograr una comprensión más profunda de los conceptos del dominio expresados mediante una notación gráfica estándar, permitiendo acercarse a nuevas tecnologías que reemplazan los diagramas o dibujos en papel (en papel, posiblemente cueste mucho modelar detalles específicos de modelado por no contar con todos los posibles elementos de modelado).

b) reducir malas interpretaciones del modelo utilizando la estricta notación y conociendo las restricciones del lenguaje a través de la experimentación de los modelos con la herramienta.

Uno de los beneficios más relevantes de estas herramientas es la navegabilidad entre los distintos diagramas que se generan durante el modelado de software con el valor agregado de la trazabilidad entre modelos la cual le permite al modelador una clara visualización de las relaciones entre los distintos artefactos producidos y la evolución de los modelos en el tiempo. Estas características nos permiten realizar un análisis previo del impacto que produciría modificar alguna regla del negocio u otro artefacto de modelado en el resto del sistema.

3. Marco de Enseñanza propuesto

En el siguiente apartado se presentan distintas cuestiones sobre la enseñanza de la IRBM y sus buenas prácticas en Educación superior. Finalmente, en la última subsección se muestra la utilidad y aporte de la construcción de modelos en la IRBM a través de la concepción del modelo CIM en etapas tempranas del desarrollo de software.

3.1. Enseñanza de la Ingeniería de Requerimientos Basada en Modelos

La Ingeniería de Requerimientos (IR) es considerada hoy una parte fundamental de la IS que cuenta con su propio campo de trabajo y avanza continuamente. El proceso de IR es un proceso iterativo que consta a su vez de tres grandes subprocesos: elicitación, especificación y validación de requerimientos [13, 14, 15]. La IR logró expandirse para no sólo abordarse en etapas tempranas del ciclo de vida del software, sino en prácticamente todas las etapas del desarrollo de sistemas. A nuestro entender, el principal motivo del avance de esta disciplina de la IS tiene que ver con el hecho de que requerimientos bien especificados y rastreables, entre los artefactos o productos de desarrollo, ayudan a una eficiente gestión del cambio, evolución y mantenimiento del producto software, llegando así a productos de mejor calidad y escalables.

Le especificación funcional puede hacerse tanto con lenguaje natural, utilizando reglas, formatos y plantillas de documentación, o bien a través de modelos gráficos construidos con lenguajes de modelado como UML y SysML. La especificación de requerimientos en lenguaje natural más ampliamente aceptada es la propuesta por IEEE [16] que sigue una plantilla standard. Los casos de uso también se utilizan como parte de la especificación de requerimientos funcionales. La documentación de los Casos de Uso responde a modelos textuales que especifican requerimientos también en lenguaje natural. Los casos de uso plantean un escenario principal y opcionalmente otros secundarios en los que un actor externo interactúa con el sistema.

La Ingeniería de Requerimientos Basada en Modelos, que es parte de la Ingeniería de Software Basada en Modelos, apunta a la conceptualización y especificación de los requerimientos exclusivamente con modelos [17]. La descripción de requerimientos con modelos más utilizada en la industria del software, es la especificación de escenarios de los casos de uso como Diagramas de Interacción de UML, más exactamente, Diagramas de secuencia UML y Diagramas de actividades UML. Los requerimientos, inevitablemente, siempre están expuestos a cambios. Vale destacar que la experiencia del equipo de trabajo en proyectos de software juega a favor para estabilizar los requerimientos en etapas tempranas de la construcción y evitar cambios de alto impacto durante el desarrollo y sobre todo al final del mismo.

De todos modos, los requerimientos cambiarán durante el desarrollo y la etapa de mantenimiento del producto software. Este hecho es natural y es de esperarse, algunas de las razones son: la naturaleza humana de los usuarios, la evolución y maduración de los requerimientos, los cambios y las reingenierías de procesos en las organizaciones, los cambios externos que no pueden ser siempre previsibles, entre otros.

Esta disciplina (IRBM) se posiciona en un lugar muy importante en los proyectos de software, especialmente en aquellos proyectos robustos, con alto nivel de complejidad y de construcción a largo plazo, debido a una serie de aspectos como:

- Trazabilidad de requerimientos entre modelos.
- Practicidad de versionado de los modelos para validación con los usuarios.
- Reuso de artefactos de modelado en la creación de patrones funcionales del dominio.
- Navegación desde los requerimientos hacia sus correspondientes artefactos de implementación (clases, tablas, decisiones de diseño, interfaz de usuario, entre otros.) y a la inversa.
- Gestión de mantenimiento y registro de cambios de los artefactos de modelado en un proyecto que evoluciona a través de sus distintas iteraciones de proceso.

3.2. Tecnologías CASE para Enseñanza de IRBM

En la diversidad y análisis comparativos de tecnologías CASE [23, 24] modernas (open source y propietarias) para: edición de modelos, transformación de modelos, validación y verificación de modelos, despliegue de aplicaciones dirigidas por modelos, entre otras clasificaciones, este trabajo utilizó la herramienta Enterprise Architect (de licencia propietaria por Sparx Systems) para el desarrollo del Project Model, tomando en consideración las siguientes características deseables para la enseñanza de IRBM y la evolución de los artefactos generados en el ciclo de vida del paradigma MDD:

- es actualmente una de las herramientas más robustas y utilizadas en la industria de software para construcción de

modelos complejos que requieren el uso de distintos lenguajes de modelado como UML, SysML, BPMN, Diagramas Entidad-Relación, entre otros.

• las transformaciones de modelos PSM a Código son fácilmente manipulables y realizables, lo que permite al alumno comprender que todo el esfuerzo realizado en etapas de la ingeniería de requerimientos no genera simplemente documentación, sino que éstos serán modelos productivos en las siguientes etapas del desarrollo.

• ofrece versionando del Project Model de manera muy intuitiva.

• permite trabajar con roles de usuarios, lo que permite vislumbrar al alumno acerca de los modelos que puede manipular un Analista de Negocio y cuáles un Diseñador de Software, y cuáles un Tester, entre otros.

• ofrece vistas interactivas para navegar artefactos de modelado entre distintos diagramas, dando lugar a que los alumnos comprendan aspectos de evolución e integración de modelos en un mismo proyecto de modelado.

• ofrece vistas que ilustran la trazabilidad entre artefactos de modelos de distintas disciplinas, dando lugar a que los alumnos comprendan el impacto de los cambios de ciertos artefactos (por ejemplo: un casos de uso o una regla de negocio) en sus modelos de realización (por ejemplo: clases, objetos de la base de datos, modelo de interfaz de usuario, entre otros).

• ofrece una vista de matriz de relaciones, que da lugar a comprender la completitud e integración de los requerimientos a través de esquemas gráficos.

• ofrece mecanismos para estimar esfuerzo de trabajo en base a los casos de uso definidos en el proyecto de modelado, dando lugar a que los alumnos comprendan que los modelos no son meramente documentativos sino que son insumos productivos para otras disciplinas, y por consecuencia dar importancia del esfuerzo y precisión en el modelado.

4. Desarrollo de un Caso de Estudio aplicando IRBM y Herramientas CASE

Como parte de la propuesta se presenta un Proyecto de Modelado, basado en un dominio de negocio paradigmático, bien conocido por la comunidad académica, como lo es el Sistema de Punto de Venta que ofrece el autor Craig Larman en su libro UML y Patrones [4].

Como producto destacable se obtuvo el modelo CIM en el ambiente MDD. El Proyecto de Modelado está dividido en 3 grandes disciplinas - Administración, Requerimientos y Análisis - organizadas y clasificadas según los modelos evolutivos de la propuesta. La distribución está dada por medio de Vistas y Paquetes, entre los cuales se detallan cada una de estas disciplinas, ilustrada en la Figura 4.

Cada una de las disciplinas juega un rol en el proceso evolutivo:

• Administración: se utiliza para llevar adelante el control de cambios que se producen en las diferentes etapas del ciclo de vida de desarrollo y la correspondiente documentación.
• Requerimientos: se determinan los servicios que el cliente requiere de un sistema, las restricciones sobre las cuales será desarrollado y se analiza su evolución; adoptando las metodologías de la IRBM, conceptualizando y especificando los requerimientos exclusivamente con modelos.
• Análisis: se descubren y explotan aquellos elementos del dominio de la problemática en cuestión.

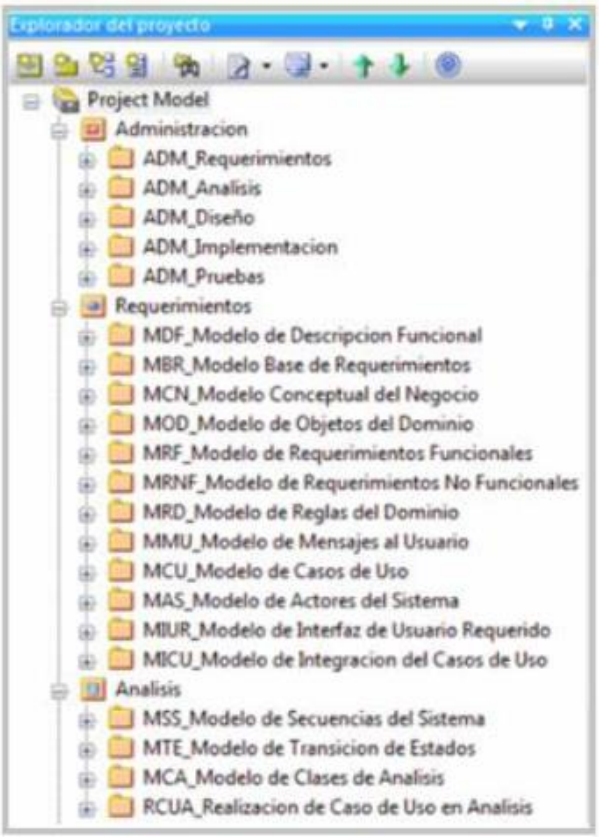

Figura. 4: Organización y Clasificación de modelos en Project Model de la herramienta CASE

4.1. Concepción CIM desde la Ingeniería de Requerimientos Basada en Modelos

Durante la etapa de Requerimientos, generamos inicialmente el Modelo de Descripción Funcional (MDF), este nos permite definir los subsistemas o módulos que forman parte del sistema, a través de textos que describen las funciones que debe cumplir el mismo.

Luego, podemos realizar el Modelo Base de Requerimientos (MBR) que contendrá los requisitos de usuario usando notación gráfica provista por SysML para modela cada uno de los subsistemas.

Continuamos con la confección del Modelo Conceptual de Negocio (MCN) que describen los procesos del negocio mediante diagramas de actividades UML. El MCN nos permite reconocer objetos esenciales del negocio que quedarán contenidos dentro del Modelo de Objetos del Dominio (MOD), los cuales luego son tomados como base en el análisis OO.

El Modelo de Requerimientos Funcionales (MRF) nos muestra cómo debería comportarse el sistema frente a determinadas entradas y situaciones particulares.
Por otra parte, el Modelo de Requerimientos No Funcionales (MRNF) representa los límites en cuanto a la operatividad del sistema, como restricciones de tiempo de respuesta, concurrencia, seguridad, tiempo máximo de respuesta, entre otros.
Las Reglas del Dominio (RD) especifican características propias del negocio. Siguiendo con la descripción de esta etapa tenemos el Modelo de Interfaz de Usuario Requerido (MIUR). Aquí se construyen prototipos de pantallas del sistema en las que sólo se modela el diseño de la Interfaz de Usuario. También se elabora el MMU (Modelo de Mensajes de Usuario) en la medida que necesitamos explicitar interacción del sistema con el usuario, desde escenarios de un Caso de Uso (CU). Una vez determinados los MBR, MCN, MOD, MRF, MNRF, MRD, MMU y MIUR comenzamos con el Modelado de Casos de Uso (MCU). Estos definen una unidad clara de funcionalidad, debiendo cumplir con un objetivo concreto y retornar un resultado de valor al usuario. Para cada CU se describe un conjunto de secuencias en donde cada una representa la interacción del sistema con los elementos externos del mismo [4,11]. Estos elementos son actores que se representan en el Modelo de Actores del Sistema (MAS).

4.2. Construyendo modelos CIM desde un enfoque Análisis Orientado a Objetos

El análisis orientado a objetos se sustenta en los artefactos de modelado producidos en la etapa de Ingeniería de Requerimientos. El modelo CIM será descrito a partir de cuatro modelos fundamentales producidos en esta etapa: El Modelo de Secuencia del Sistema (MSS), Modelo de Clases Análisis (MCA),
Modelo de Transición de Estados (MTE) y Realización de Casos de Uso en Análisis (RCUA). Cada uno permitirá a los alumnos ver y desglosar un modelo inicial independiente de los términos computacionales en distintas perspectivas. El MSS define, para un escenario particular de un CU, los eventos que los actores generan, su orden y los eventos que se intercambian entre sistemas.

En el MCA se identifican y visualizan las relaciones entre las clases conceptuales que involucran el sistema, diseñando en base a objetos derivados de la etapa inicial (MOD y MCU).

El MTE se utiliza para acoplar y aportar mayor información a aquellas clases del MCA en las que es posible reconocer distintos estados. Esto lo realiza a través de los diagramas de máquinas de estados UML que permiten mostrar el comportamiento interno de un elemento a través de sus estados internos y sus transiciones.

Con el fin de interrelacionar todos los artefactos independientes por cada CU descritos en la etapa anterior, se aplica el modelo de RCUA, proporcionando una vista integradora de lo producido en esta etapa centrado en el CU.

De este modo podemos empezar a observar la evolución que sufre el CU al pasar de la etapa de requerimientos a la disciplina de análisis. Hasta aquí, con la realización de todos los modelos de requerimientos y de análisis se ha completado el producto CIM en contexto MDD, insumo para construir el PIM.

5. Aportes de la IRBM en apoyo al Auto aprendizaje del paradigma MDD

Dado que la construcción del CIM es la etapa más artesanal del paradigma MDD [1], porque se construyen modelos del dominio del negocio, elicitando y especificando los requerimientos, con modelos independientes de la plataforma y del paradigma de programación, hemos optado por la enseñanza y aplicación práctica en el ámbito académico, de la metodología IRBM para la construcción del modelo CIM en el Proyecto de Modelado en apoyo a la enseñanza de MDD.
Las etapas de la IRBM exige un nivel de organización conceptual del mundo real y en este contexto, la adopción de su tratamiento bajo un enfoque de modelado con distintos niveles de abstracción, es una adecuada elección para que puedan ir incrementando complejidad a los modelos en la medida que van evolucionando en la captura de conocimiento y talleres de requerimientos con los usuarios o stakeholders de la organización en cuestión.
Los aspectos del desarrollo evolutivo de los modelos, que especificarán los requerimientos de sistema, aporta el valor del auto aprendizaje continuo a los alumnos, ya que deben verificar y validar cualitativamente los modelos ya construidos, para luego avanzar sobre los nuevos modelos. Este ejercicio ofrece un espacio natural para mejorar o refinar los modelos actuales, antes de avanzar en los modelos siguientes (ya sean modelos gráficos o textuales).

En esta propuesta nos limitamos a poner en práctica la construcción de modelos CIM en MDD, por lo que las cuestiones referidas a transformaciones automáticas entre modelos quedan exceptuadas y fuera de alcance.

Es fundamental para MDD que los modelos CIM, desde los que se aplica la transformación automática de modelos hacia modelos PIM, sean robustos y completos.

Para lograr esto último se aplican buenas prácticas de IRBM junto con lenguajes de modelado UML y OCL para eliminar ambigüedades. En la Figura. 5 se muestran los grandes tópicos tratados en el Proyecto de Modelado.

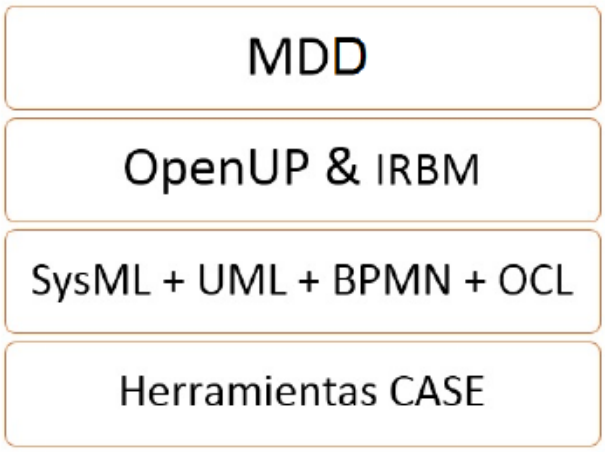

Figura. 5: Conceptos y estándares aplicados en el Proyecto de Modelado

El uso de herramientas CASE aporta un valor importante a los alumnos, al momento de evaluar las reglas de buena formación de los diagramas. Por tanto, el alumno no solamente aprende a modelar, sino que también aprende las reglas de buena formación de ciertos modelos gráficos. Por otro lado, los alumnos valoran que los modelos no son puramente planos o esquemas gráficos, sino que a partir de éstos, es posible explotar la información de estos artefactos de modelado para salvar inquietudes clásicas de la ingeniería de software, tales como: análisis de alto o bajo acoplamiento de modelos, visualización de la matriz de
requerimientos con artefactos de modelado de otras disciplinas como Diseño, Implementación o Pruebas.
Finalmente, luego que los alumnos han trabajando en actividades rigurosas de modelado, han manejando distintos niveles de abstracción, han logrado concebir la importancia de asegurar trazabilidad entre los artefactos de modelado, evaluar impactos en distintos artefactos de modelado ante modificaciones en los requerimientos, entre otros, los alumnos ya se encuentran en condiciones de incorporar conocimientos avanzados del tema. Es aquí donde hemos logrado las bases teóricas-prácticas iniciales para explorar conceptos avanzados del paradigma MDD, tales como: modelos y metamodelos, lenguajes de transformación de modelos, herramientas de transformación, configuraciones más detalladas en el ciclo de vida MDD, entre otros.

6. Resultados y experiencia obtenida hasta el momento

La propuesta presentada, aplicando las buenas prácticas de IRBM haciendo uso de tecnología CASE, es una propuesta que guía la enseñanza del paradigma MDD. Permite introducir a los alumnos en una nueva forma de pensar y construir Sistemas de Software. La base de la propuesta se sustenta en tecnología CASE, debido a que si la escritura de los modelos se realizara en forma manual, contando con un número considerable de elementos y diagramas, trazar un vínculo entre estos sería una actividad casi imposible, haciéndolos incomprensibles e inconsistentes.

La propuesta de este trabajo fue implementada por primera vez, durante el ciclo lectivo del año 2010, en el marco del dictado de la asignatura Ingeniería de Software, correspondiente al 4to año de la carrera de Ingeniería en Sistemas de Información, en la Universidad Tecnológica Nacional - Facultad Regional La Plata, con un curso de 35 alumnos, divididos en equipos de trabajo de tres alumnos por equipo, para llevar adelante trabajos de modelado e inspección del Project Model (ya resuelto) que la cátedra ofreció para evaluar ciertos aspectos de manejo de la tecnología CASE utilizada. Estos alumnos poseen conocimientos previos básicos adquiridos en la asignatura Paradigmas de Programación de 2do año de la carrera, sobre lenguajes de modelado estándar; sobre Análisis de Sistemas, que es una asignatura del mismo año; y sobre y Diseño de sistemas, que es de 3er año.

En este contexto, la prueba de enseñanza arroja una evaluación de resultados de esta propuesta, que se realizó ejecutando un coloquio a cada equipo de trabajo, realizando preguntas bien específicas sobre un Project Model que ofreció la cátedra, para realizar inspecciones de comprensión cualitativa de los modelos presentados y al mismo tiempo descubrir la capacidad para navegar artefactos de modelado, precisar impactos en la modificación de ciertos modelos, crear y eliminar trazas, entre otros aspectos evaluativos.

Los materiales utilizados son una PC con la aplicación EA Trial (versión gratuita por un mes) instalada y un proyector para mostrar al grupo docente las consultas planteadas en el coloquio.

La realización del coloquio apoyado en la inspección de un Project Model sobre Herramientas CASE resultó en más del 80 % positiva, dado a que los alumnos debían haber tomado ejercicio en el manejo de la herramienta CASE para resolver las consultas precisas, por tanto el coloquio no les resultó difícil. El porcentaje restante no superó la experiencia dado a que no tenían mucha práctica

sobre modelado ni tampoco habían inspeccionado el Project Model que propuso la cátedra.
Esta motivación, nos permite dar a conocer a los alumnos la existencia de otro tipo de tecnologías de desarrollo bajo esta concepción paradigmática, separándolos de las tradicionales, y mostrar que se pueden realizar estas tareas de manera sencilla.

El uso de estas herramientas mejora la productividad en el desarrollo de software reduciendo el coste en términos de tiempo y de dinero, aportando además muchas facilidades para el modelador en lo que respecta a edición y documentación de artefactos de modelado, evolución de modelos a través de la gestión de cambios en construcción y documentación interna, navegación a distintos diagramas de manera automática, vista actualizada de las relaciones físicas entre artefactos de modelado y esquema de trazabilidad en la construcción de modelos.

El Project Model propuesto favorece la enseñanza del paradigma MDD, ya que por medio de la herramienta CASE, se motiva y ofrece el uso de los medios que nos provee, permitiendo ver las relaciones entre artefactos, la matriz de relaciones y la trazabilidad entre los modelos.

Como podemos observar en la Figura 6, el Modelo Base de Requerimientos (MBR) debe darle soporte a los requerimientos diagramados en el Modelo de Requerimientos Funcionales (MRF); definiendo el MBR, los límites para el modelado en etapas posteriores.

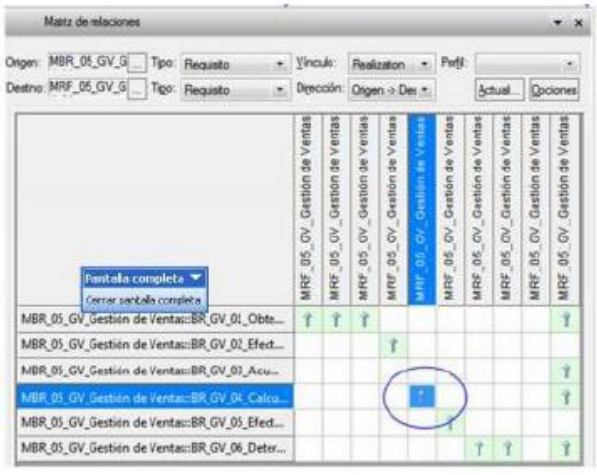

Figura 6: Matriz de relaciones entre artefactos interrelacionados: evaluación de impactos.

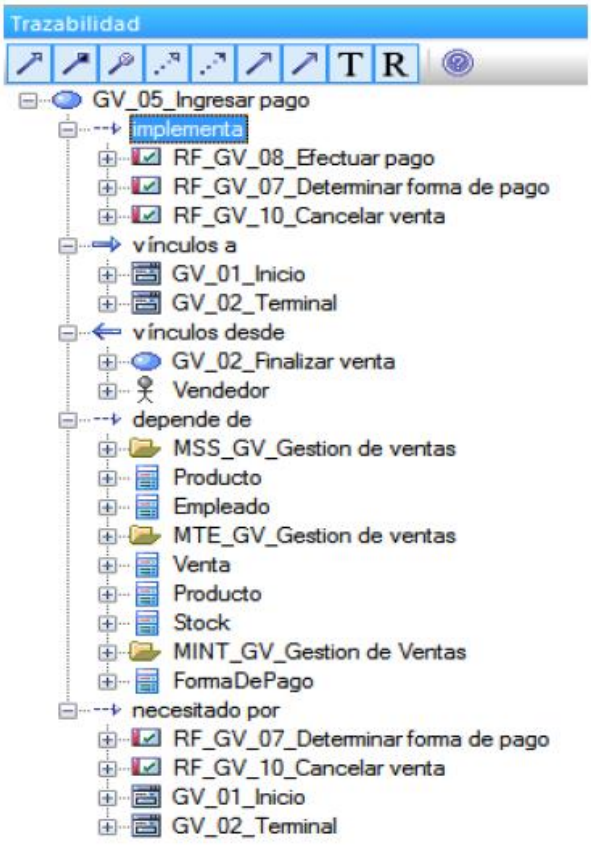

Figura 7: Jerarquía de artefactos para evaluación de impactos por cambios de especificación.

La ventana de trazabilidad (Figura 7) permite ver la evolución del artefacto desde sus orígenes, el sustento que le brinda a otros elementos de modelado, los vínculos que se generan y todas descripciones a distintos niveles de abstracción.

Por otro lado, la ventana de Relaciones (Figura 8) permite visualizar todos los artefactos de modelado que tienen conexión con un artefacto en particular.

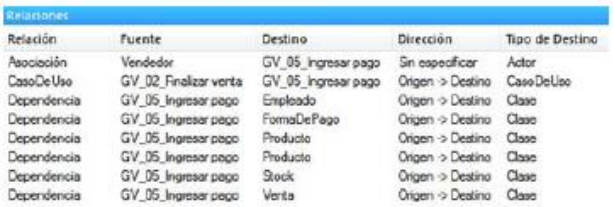

Figura 8: Relaciones con el foco en un artefacto concreto.

Conclusiones y Trabajo Futuro

Si bien hemos podido observar cambios positivos en contra de la reticencia que los alumnos demuestran frente a cambios bruscos de paradigmas tecnológicos, todavía éste sigue siendo un tema complejo dada la renovación radical que se detecta en la forma de producir código

automáticamente. Se nota en el alumnado, el interés que genera la idea de lograr obtener código compilable desde modelos, en forma automatizada, haciendo desaparecer virtualmente la figura tradicional del programador. A su vez, el hecho de no profundizar hacia un proceso completo dirigido por modelos (o sea, llegar a obtener el código desde sucesivas transformaciones a partir de un modelo PIM) deja incompleta la idea principal que se pretende introducir acerca de este nuevo paradigma. Sin embargo, hemos comprobado que introducir las ideas principales del proceso tradicional basado en modelos, mediante el uso de herramientas CASE y técnicas de ingeniería de requerimientos, para lograr producir los modelos CIM y PIM a través de un proyecto de modelado, facilita la transición a la adopción final de MDD, a la vez que deja explicita la reusabilidad y utilidad de mucho de los conceptos ya estudiados sobre la nueva tecnología MDD.

El marco teórico y metodológico presentado en este trabajo, producto de la experiencia en la enseñanza de estos temas [19,20], brinda una alternativa pedagógica para la integración de contenidos que generalmente quedan fuera de la currícula. Mediante la utilización de conceptos y contenidos más familiares para los estudiantes de sistemas de información, se amplía la concepción rígida que involuntariamente se imprime durante su carrera sobre el proceso de desarrollo de software.

De esta manera, en el ámbito enseñanza/aprendizaje, tanto en el aula, a través del docente, como virtualmente al utilizar el alumno herramientas software con asistencia propia, se conectan los conocimientos sobre el diseño de software y sus técnicas, en conjunto
con las técnicas de la Ingeniería de Requerimientos, para introducir la importancia del modelado, cumpliendo con las primeras etapas de un desarrollo MDD.

Además, se destaca el ejercicio pedagógico de promover autoevaluación de aprendizajes [21,22] y la forma de introducir al alumno conceptos de gerenciamiento de proyectos desde un enfoque conducido por modelos.

Como trabajo futuro planteamos un esquema de trabajo similar para la enseñanza del campo BPM (Business Process Management) utilizando el lenguaje de modelado BPMN (Business Process Modeling Notation) con el objeto de modelar los procesos de negocio de un caso de estudio para llegar a un modelo CIM incluyendo aspectos de negocio en etapas tempranas del proceso MDD. Y en avance de este último, promover que los alumnos definan reglas de transformación de modelos (utilizando lenguajes de transformación para tal fin) utilizando como base el mismo Caso de Estudio propuesto en este trabajo, en el Project Model.

Agradecimientos

Esta comunicación es resultado de los avances del Proyecto de I&D "Modelado Ágil para Producción de Software en MDD" homologado por Programa de Incentivos del Ministerio de Educación de la Nación Argentina y financiado por la SCTyP del Rectorado UTN. También agradecemos la colaboración de los ayudantes de cátedra e investigadores-alumnos Lautaro Mendez y Javier Marchesini, en la revisión del proyecto de modelado y actividades pedagógica de puesta en ejecución para validar esta propuesta en el ambiente académico universitario.

Referencias

[1] Pons Claudia, Giandini Roxana y Pérez, Gabriela, Desarrollo de Software Dirigido por Modelos. Conceptos teóricos y su aplicación práctica, EDULP & McGraw-Hill, 1er. Edición, (2010).
[2] Pineda C., Un método de desarrollo de Hipermedia Dirigido por Modelos, Tesis Doctoral, Universidad de Valencia, Diciembre 2008.
[3] J. Rumbaugh, I. Jacobson, Grady Booch. El Lenguaje Unificado de Modelado, Manual de Referencia. Addison Wesley, Primera Edición, 2000.
[4] Craig Larman - UML y Patrones - 2da Edición - Prentice Hall - 2003 - ISBN: 84-205-3438-2.
[5] Herrera, Giovanni. Integración de UML y Lenguajes de Modelado Específicos de Dominio mediante la generación automática de perfiles UML. Tesis Doctoral. Universidad Valencia. Septiembre 2006.
[6] Tim Weilkiens, Systems engineering with SysML/UML: modeling, analysis, design.
[7] Friedenthal & Moore & Steiner A practical Guide to SysML: The Modeling Language, Morgan Kaufmann, Second Edition, 2011, ISBN: 9780123852069
[8] SysML: http://www.sysml.org/docs
[9] Jos Warmer and Anneke Kleppe. The Object Constraint Language. Getting Your Models Ready for MDA. 2nd Edition. Addison-Wesley Professional. (2003). ISBN-13: 978-032-11-7936-4
[10] OCL: http://www.omg.org/spec/OCL/
[11] Grady Booch, James Rumbaugh y Ivar Jacobson. El lenguaje unificado de modelado. Segunda Edición. Pearson Education, S.A. (2006). ISBN-13: 978-847-82-9076-5
[12] Proceso OpenUP. www.eclipse.org/epf/general/OpenUP.pdf
[13] Loucopoulos, P, Karakostas, V, System Requirements Engineering, McGraw Hill, 1995.
[14] Ian Sommerville, Ingenieria de Software, Séptima edición, Pearson Educacion S.A., 2005.
[15] Roger Pressman. Ingeniería del Software un enfoque práctico. Sexta Edición. Mc Graw-Hill. (2005).
[16] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements.
[17] DeMarco, Tom. Structured Analysis and System Specification. Englewood Cliffs, NJ, 1979.
[18] Enterprise Architect:www.sparxsystems.com.au
[19] Figueroa, N., Cataldi, Z., Méndez, P., Rendon,J., Costa, G., Salgueiro, F., Lage, F. Los estilos de aprendizaje y el desgranamiento universitario en carreras

de Informática. Jornadas de Educación en Informática y TICs en Argentina (JEITICS).
[20] Rosanigo, Z., Paur, A., Bramati, P. Tecnología Informática Aplicada en Educación. Jornadas de Educación en Informática y TICs en Argentina (JEITICS). RedUNCI. 2005.
[21] Stella M. Vázquez. Rendimiento Académico y Patrones de Aprendizaje en Estudiantes de Ingeniería. Revista Ingeniería y Universidad. Print version ISSN 0123-2126 - Vol.13 N° 1 - Bogotá (2009).
[22] Yanko O. Núñez y Víctor B. Núñez. Evaluación de los Aprendizajes en Ingeniería. Revista Facultad de Ingeniería, U.T.A. Chile, Vol.11 N°1. 2003.
[23] J. Quintero y R.Anaya. Marco de Referencia para la Evaluación de Herramientas Basadas en MDA. 2007. X° Conferencia Iberoamericana de Software Engineering. ISBN 978-980-325-323-3
[24] J. Quintero, R. Anaya, J. Marín, A. Bilbao López.Un estudio comparativo de herramientas para el modelado con UML.2005. Revista Universidad EAFIT.
[25] Eclipse Process Framework Project (EPF). http://www.eclipse.org/epf/

Dirección de Contacto de los Autores:

Ing. Leopoldo Nahuel
Av. 60 esquina 124 s/n – CP 1900
La Plata, Provincia de Buenos Aires, Argentina
lnahuel@frlp.utn.edu.ar

Ing. Cecilia Ariste
Av. 60 esquina 124 s/n – CP 1900
La Plata, Provincia de Buenos Aires, Argentina
cariste@frlp.utn.edu.ar

Dra. Roxana Giandini
Calle 50 y 120 s/n – CP 1900
La Plata, Provincia de Buenos Aires, Argentina
giandini@lifia.info.unlp.edu.ar

Leopoldo Nahuel es Ingeniero en Sistemas por UTN y Maestrando en Ingeniería de Software por UNLP. Docente de Diseño de Sistemas. Investigador y dirige Proyectos de I&D en áreas de Desarrollo Dirigido por Modelos en LINSI. Trabaja en la industria en Ingeniería de Procesos & Software.

Cecilia Ariste es Ingeniera en Sistemas por UTN y Maestrando en Ingeniería de Software por UNLP. Docente de Análisis de Sistemas. Investigador en áreas de Ingeniería de Requerimientos en LINSI. Trabaja en la industria en Análisis Funcional en proyectos de desarrollo de software.

Roxana Giandini es Doctora en Ciencias Informáticas por UNLP. Autora del libro "Desarrollo de Software Dirigido por Modelos. Conceptos teóricos y su aplicación práctica". Director de Proyectos I&D en áreas de métodos formales en LINSI/LIFIA. Asesora en Ciencia y Tecnología en UTN.