Construcción de modelos CIM/PIM en MDA aplicando Ingeniería de Software Basada en Modelos, lenguajes notacionales SysML-UML-OCL y herramientas CASE de soporte a la evolución y trazabilidad de modelos

Di Girolamo Romina, Girado Pablo, Mendez Lautaro, Perelli Julián, Vargas Lucas

Universidad Tecnológica Nacional – Facultad Regional La Plata – Dpto. de Sistemas
Laboratorio de Innovaciones en Sistemas de Información, LINSI
{rdigirolamo, pgirado, lmendez, jperelli, lvargas }@linsi.edu.ar

Resumen: El Desarrollo Dirigido por Modelos (MDD) se ha convertido actualmente en un importante paradigma de la Ingeniería de Software, proponiendo sustituir - como artefacto principal en el proceso de desarrollo del software - al código fuente de lenguajes de programación por modelos, permitiendo nuevas posibilidades de crear, analizar y manipular sistemas. En este ámbito, los aspectos de rastreo o trazabilidad son un importante desafío teórico-práctico, necesarios tanto en actividades de modelado manual como en procesos de transformación automática entre modelos. Proponemos integrar - a través del desarrollo práctico de un Proyecto de Modelado - las bases conceptuales de MDD, técnicas de Ingeniería de Software Basada en Modelos, lenguajes notacionales SysML-UML-OCL, buenas prácticas del proceso iterativo RUP y uso de herramientas CASE, promoviendo mecanismos que destacan evolución y trazabilidad entre artefactos de modelado durante el proceso de construcción de modelos CIM/PIM, en contexto del estándar MDA para la visión MDD.

Palabras claves: Ingeniería de Software Basada en Modelos (ISBM), Desarrollo Dirigido por Modelos (MDD), Arquitectura Dirigida por Modelos (MDA), Lenguajes de modelado, Proceso Unificado de Rational (RUP), Herramientas CASE, Trazabilidad de modelos.

1 Introducción

Los modelos son representaciones parciales de los sistemas que describen de forma rigurosa ciertos aspectos, en contraposición al lenguaje natural, utilizando una notación y una semántica claramente definida. Éstos son parte fundamental del paradigma de desarrollo de software conocido como Model Driven Development (MDD) [1].
En las etapas de MDD se parte de los requisitos del sistema y se desarrollan modelos de arquitectura/diseño (CIM, PIM y PSM) desde niveles de abstracción altos a niveles más detallados y concretos, que incluyen la definición de la plataforma de ejecución.

El consorcio OMG (Object Management Group) [2] propone un estándar para el paradigma MDD denotado por sus siglas MDA (Model Driven Architecture) [3] el cuál se sustenta en tres lenguajes notacionales estándar: UML (Unified Modeling Language) [4], SysML (Systems Modeling Language) [5] y OCL (Object Constraint Language) [6]. Éstos estándares mencionados brindan una base formal para la generación de herramientas CASE [7], las cuales asisten el desarrollo de proyectos siguiendo buenas prácticas de la Ingeniería de Software Basada en Modelos [8] y suministrando funcionalidades específicas para dar soporte a MDA, como la transformación entre modelos y generación automática de código fuente.
En la última década ha surgido una corriente metodológica de soporte al desarrollo evolutivo de aplicaciones de software [9], centrada en la productividad e integración de los equipos de trabajo para lograr una pronta interacción con el usuario a través de iteraciones discretas de tiempo, permitiendo así validar y consensuar los requerimientos de la aplicación requerida de forma temprana. Resulta entonces interesante poder combinar o relacionar la metodología MDA con los procesos iterativos para la construcción de software.
En el contexto MDA, la trazabilidad o rastreo de elementos es un importante desafío teórico y práctico [10] [11] [12]. Estos mecanismos de rastreo son necesarios en procesos que involucran transformación automática entre modelos, en generación de código y en actividades del proceso de modelado manual.
Por lo expuesto, el objetivo general de este artículo es abordar el estudio, análisis y desarrollo de un Proyecto de Modelado desde los Requerimientos de Negocio hasta el Diseño Orientado a Objetos detallado, generando modelos precisos CIM/PIM en un ámbito MDA, siguiendo lineamientos metodológicos, buenas prácticas del proceso RUP [13] y utilizando herramientas CASE.

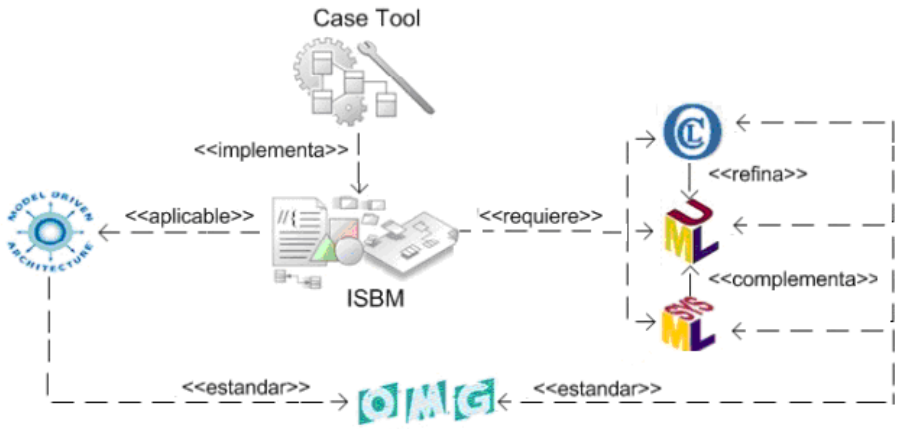


Fig.1. Bases conceptuales en apoyo a la Ingeniería de Software Basada en Modelos (ISBM)

Cada uno de los elementos mostrados en la Fig. 1, serán descriptos brevemente en las siguientes secciones con el objetivo de dar contexto y apoyo a los estándares empleados en el desarrollo práctico del Proyecto de Modelado que detallaremos al final de éste artículo.

2 Marco de trabajo por OMG a la ISBM y al paradigma MDD

El desarrollo de software dirigido por modelos conocido por sus siglas MDD es un paradigma que sitúa a los modelos como entidades fundamentales que dirigen el proyecto de construcción de software de principio a fin. Puede considerarse como una evolución automatizada para la ISBM, ya que al contrario del enfoque tradicional donde se consideraba como elemento principal al código, en MDD los modelos tienen igual importancia que el código mismo. Los modelos - desde los más abstractos a los más concretos - se generan aplicando transformaciones hasta llegar al código fuente. La característica principal es que las transformaciones son generadas por computadoras, librando de esta tarea a las personas, logrando de este modo convertir a los modelos de entidades estáticas (que eran interpretadas por analistas y diseñadores) a entidades productivas dentro del proceso de desarrollo.

Las bases que sostienen al paradigma MDD son:

- Abstracción: definiendo lenguajes de modelado específicos para cada dominio y se ocultan aspectos relacionados con las tecnologías de implementación. Estos utilizan una sintaxis clara que resulta amigable y transmiten los conceptos del dominio, reduciendo la dependencia que existe entre los modelos y la tecnología, permitiendo que dichos modelos puedan ser implementados en múltiples plataformas.
- Automatización: es la forma más eficaz para aumentar la productividad y la calidad. MDD hace uso de las computadoras para mecanizar tareas repetitivas, evitando que los humanos las realicen. La automatización incluye tareas como: transformación de modelos expresados mediante conceptos de alto nivel, específicos del dominio, en su correspondiente código fuente que será ejecutado en una plataforma concreta.
- Estándares: debe ser implementado mediante estándares abiertos, aportando numerosos beneficios como la posibilidad de intercambiar especificaciones entre herramientas complementarias o equivalentes de diferentes proveedores.

MDD identifica distintos tipos de modelos, bien conocidos por sus siglas en ingles:

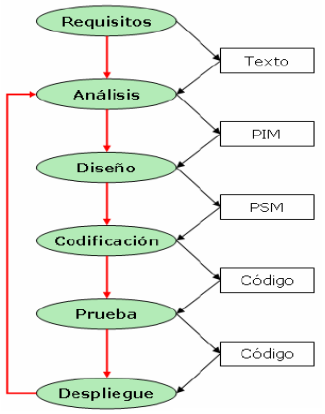
CIM (*Computational Independent Model*): describen la lógica del dominio del negocio desde una perspectiva independiente de la computación.

PIM (*Platform Independment Model*): describen de forma abstracta la funcionalidad del sistema de forma independiente a cualquier tecnología de implementación.

PSM (*Platform Specific Model*): describen el sistema en términos de construcciones implementativas específicas ligadas a una tecnología concreta.

CODE ó **IM** (*Implementation Model*): describe o especifica el sistema en término del código fuente (modelo texto) en una tecnología concreta.

En la Fig. 2 se ilustra el ciclo de vida iterativo MDD mientras que en la Fig. 3 el proceso de transformación MDD ejemplificado con tres tecnologías concretas. Las líneas que se observan en la Fig. 3 indican los procesos automatizados mediante el uso de computadoras. Dado que el PSM es un modelo muy cercano al código fuente, la verdadera ventaja y característica clave de MDD está en pasar de PIM a PSM de manera automática.



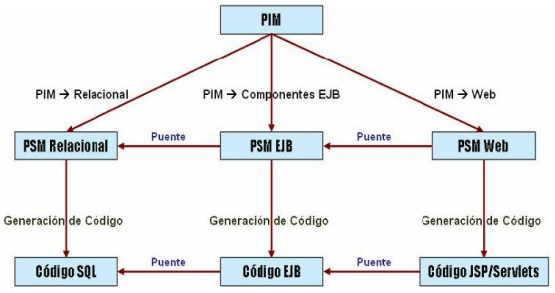


Fig. 2. Ciclo de vida MDD.

Fig. 3. Desarrollo en MDD con varios PSMs.

Es hoy día MDA (Model Driven Architecture) la propuesta concreta estandarizada por OMG que especifica los métodos y guías para el desarrollo de software, ideado para dar soporte al paradigma MDD. Una de las principales características de MDA es que separa la arquitectura, el diseño y la implementación del sistema, pudiendo tomar medidas relacionadas con el dominio del problema independientemente de la plataforma tecnológica en la cual este implementado el software. También se logra un seguimiento de la evolución del proyecto mediante modelos abstractos y tiende a aumentar la automatización mediante la transformación automática de modelos. Siguiendo un proceso MDA primeramente tiene lugar la creación de un PIM que define la funcionalidad y alcance del sistema. Es aquí que aparece un modelo no definido por MDD, el PDM (Platform Definition Model) que define una plataforma en particular, de esta manera dado un PDM como ser .NET, CORBA, web, etc., se da lugar a una transformación del PIM a un PSM. Siguiendo este proceso podemos transformar automáticamente el PIM a uno o más PSMs, lo que nos independiza de la plataforma y cambios tecnológicos [14]. Las principales motivaciones de MDA son la interoperabilidad (independencia de los fabricantes a partir de las estandarizaciones) y portabilidad (independencia de la plataforma tecnológica) de los sistemas de software. En MDA se hacen uso de lenguajes de modelado definidos y estandarizados por la OMG como son UML, OCL y SysML, los 3 adoptados para el desarrollo del Proyecto de Modelado que presentaremos mas adelante:

UML: lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Es independiente de los métodos de desarrollo, etapas del ciclo de vida, dominio de aplicación y provee fuerte apoyo a la mayoría de los proceso de desarrollo orientado a objetos.

OCL: lenguaje formal no gráfico para describir restricciones de buena formación y reglas de negocio sobre construcciones orientadas a objetos. Estas expresiones que agregan información vital a los modelos estáticos orientados a objetos y otros artefactos que modelan el comportamiento dinámico de construcciones orientadas a objetos(Figura 4).

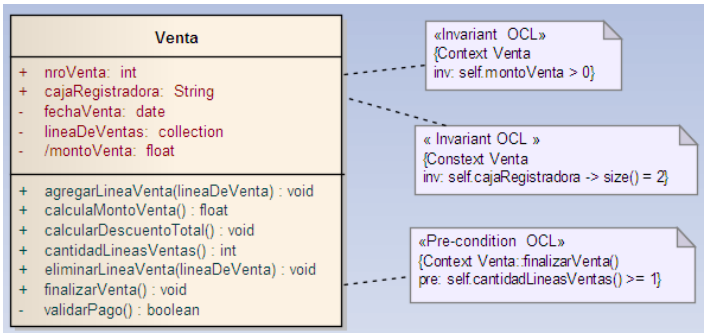


Fig. 4. clase UML enriquecida con expresiones OCL.

SysML: lenguaje de modelado de propósito general para sistemas de aplicaciones ingenieriles. Soporta la especificación, análisis, diseño, verificación y validación de una amplia gama de sistemas software y no software, o combinación de los anteriores incluyendo distribución de hardware, software, información, procesos, personal e instalaciones. SysML reutiliza un subconjunto de UML 2.0 y proporciona extensiones para dar soporte a la Ingeniería de Requerimiento. Es frecuentemente implementado como Plug-in en herramientas de modelado UML más populares.

3 Adopción de herramientas y procesos aplicable a ISBM

Existen numerosas propuestas metodológicas para el desarrollo de software que inciden en distintos aspectos del proceso de desarrollo. Entre las principales metodologías orientadas a modelos estándar podemos mencionar a RUP (Rational Unified Process) y MSF (Microsoft Solution Framework) [15], que centran su atención en llevar una documentación completa y precisa de todo el proyecto y en cumplir con un plan claro de proyecto, definido en la fase inicial del desarrollo. Una posible mejora a las propuestas tradicionales es centrarse en otros aspectos, como ser el factor humano o el producto software. Esta es la filosofía de las metodologías iterativas y ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Ejemplos de estas metodologías son: XP (eXtreme Programming), Scrum, Iconix, Cristal Methods, AUP, entre otras [1, 2].
Luego de evaluar los distintos tipos de metodologías antes mencionadas se observaron ciertas ventajas sobre la metodología RUP, entre ellas es necesario destacar que promueve el desarrollo iterativo y organiza el desarrollo de Sistemas de Software, poniendo énfasis en los requisitos y el diseño. Por otra parte RUP se ha diseñado conjuntamente con UML, por lo que la descripción del flujo de trabajo se orienta alrededor de los modelos UML asociados, razón por la cual se concluyo en la utilización de esta metodología como punto de partida. El proceso de desarrollo RUP aplica varias de las mejores prácticas en el desarrollo moderno de software en una forma que se adapta a un amplio rango de proyectos y organizaciones. Provee a cada miembro del equipo, un fácil acceso a una base de conocimiento con guías, plantillas

y herramientas para todas las actividades criticas del desarrollo de software. RUP incorpora muchas buenas prácticas en el desarrollo de software moderno, las cuáles se deben tener presentes en el desarrollo de aplicaciones para garantizar el éxito del proyecto, tales como: Desarrollo iterativo, Gestión de Requerimientos, Arquitectura basada en componentes, Modelado Visual, Verificación de la calidad en forma continua y control de cambios. Las principales ventajas de esta metodología son: mitigación temprana de posibles riesgos altos, progreso visible en las etapas tempranas, el conocimiento adquirido en una iteración puede aplicarse de iteración a iteración y los usuarios involucrados continuamente. RUP es un modelo que identifica cuatro fases bien diferenciadas como puede observarse en la Fig. 5. En cada una se realizan actividades –generando productos intermedio- en distintas disciplinas (en mayor o menor medida) en forma iterativa:

Inicio: El objetivo de la fase de inicio es el de establecer un caso de negocio para el sistema. Se deben identificar todas las entidades externas (personas y sistemas) que interactuarán con el sistema y definir estas interacciones.

Elaboración: Los objetivos de la fase de elaboración son desarrollar una comprensión del dominio del problema, establecer un marco de trabajo arquitectónico para el sistema, desarrollar el plan del proyecto e identificar los riesgos clave del proyecto.

Construcción: Esta fase comprende el diseño del sistema, la programación y la elaboración de las pruebas. Durante esta fase se desarrollan e integran las partes del sistema. Al terminar esta fase, debe tener un sistema software operativo y la documentación correspondiente lista para ser entregada a los usuarios.

Transición: La fase final del RUP se ocupa de mover el sistema desde la comunidad de desarrollo a la comunidad del usuario y hacerlo trabajar en un entorno real.

Adopción de herramientas y metodología de desarrollo aplicable a ISBM

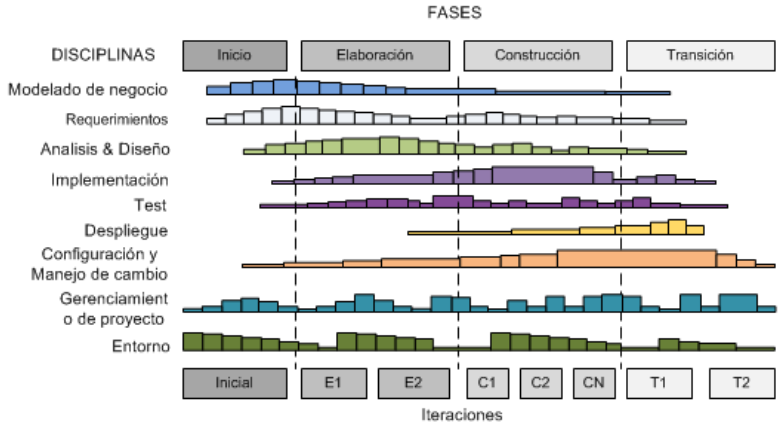


Fig. 5. Framework RUP: fases, iteraciones, disciplinas y sinergia de su ciclo de vida.

Las Herramientas CASE son un conjunto de aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en

todos los aspectos del ciclo de vida de desarrollo del software asistiendo a las tareas de análisis, diseño del proyecto, implementación del código, compilación automática, documentación o detección de errores, dentro de las características más importantes. Unos de los beneficios de éstas 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. Las herramientas permiten también realizar un estudio de estimaciones con el cual podemos definir el alcance del sistema, la cantidad de personal necesario para llevar a cabo el proyecto, el esfuerzo, los costes y la planificación que se requerirá para realizar todas las actividades y construir todos los productos asociados con el proyecto.
Durante esta investigación se evaluaron de forma equitativa herramientas CASE, tanto Open Source como propietarias. Cada una de ellas presentaron distintas características para modelar software usando sus propias funcionalidades pero se decidió optar por la herramienta Enterprise Architect [16] que da soporte a los estándares mencionados y poseer variada documentación para construir modelos.

4 Evolución y Trazabilidad en construcción de modelos CIM/PIM en un contexto MDA

De lo expuesto anteriormente se sabe que en MDA los modelos CIM/PIM deben ser precisos y completos para que las transformaciones entre modelos puedan ser realizadas de manera eficaz y automáticas. Tomando en cuenta la evolución de los modelos en las distintas fases del ciclo de vida de desarrollo de software, los lenguajes como UML, OCL, SysML y la trazabilidad lograda, por medio de las herramientas CASE, se desarrollará a continuación la construcción de un *Project Model* basado en un dominio de negocio paradigmático que persigue finalmente el producto PIM del ámbito MDA.

4.1 Características del Project Model

En la disciplina de Administración se registran los cambios (a través de la vista de mantenimiento) que se producen en las distintas etapas del ciclo de vida desarrollo de software. En la etapa de Requerimientos se determinan los servicios que el cliente requiere de un sistema, restricciones bajo las cuales será desarrollado y funcionará el mismo. En la etapa de Análisis se descubren y explotan los objetos del dominio del problema. En la etapa de Diseño se definen los objetos software y cómo colaboran para satisfacer los requerimientos. El Project Model está distribuido en paquetes en los cuales se detallan las distintas disciplinas de modelado acorde a la metodología RUP, como muestra la Figura 6.

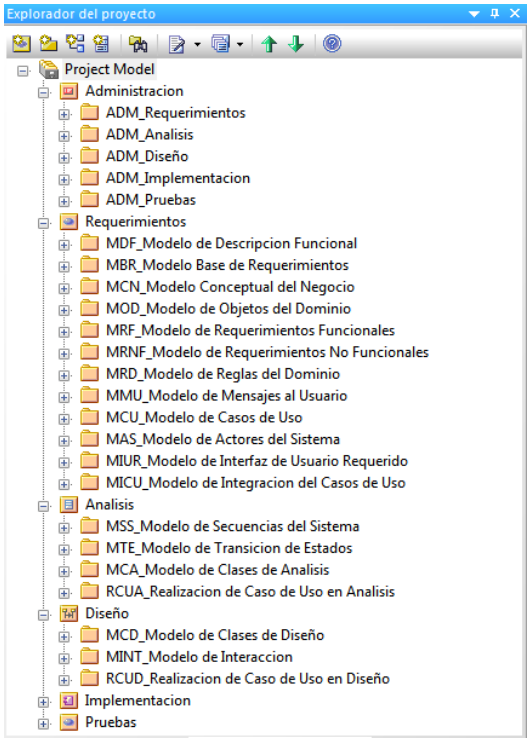


Fig. 6. Project Model

4.2 Concepción de CIM en la Ingeniería de Requerimientos basada en modelos

Durante la etapa de Requerimientos, generamos inicialmente el Modelo de Descripción Funcional (MDF), este nos permite definir de forma general los módulos o subsistemas que tendrá el sistema, a través de textos en lenguaje natural que describen las funciones que debe cumplir el mismo. Luego, podemos realizar el Modelo Base de Requerimientos (MBR) que contendrá los requisitos de usuario (expresado en lenguaje natural) usando plantillas de requerimientos sysml para cada uno de los subsistemas. Continuamos con la confección del Modelo Conceptual de Negocio (MCN) en el cual se construyen diagramas de actividades UML que describen los procesos del negocio mediante flujos de control entre actividades relacionadas. El MCN nos permite reconocer objetos esenciales del negocio que quedarán contenidos dentro del Modelo de Objetos del Dominio (MOD), para luego tomar como base en el análisis OO.
Los requerimientos son el producto fundamental de esta etapa. 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 las restricciones en cuanto a la operatividad del sistema, como restricciones de inicio, restricciones en el proceso de

desarrollo, estándares, concurrencia, seguridad, tiempo de respuesta, fiabilidad, estabilidad, tiempo máximo de respuesta, capacidad de almacenamiento, entre otros. Las Reglas del Dominio (RD) especifican características propias del negocio. Si los requisitos del Dominio no se satisfacen, en algunos (muchas) veces el sistema se puede volver impracticable.
Una vez determinados los MBR, MCN, MOD, MRF, MNRF, MRD comenzamos con el Modelado de Casos de Uso (MCU). Para el buen modelado es imprescindible definir concreta y eficazmente los componentes antes descriptos que son los que le van a dar sustento a los CUs. Los CUs definen una unidad clara y completa de funcionalidad, debiendo ser acotados (cumpliendo un objetivo concreto y retornar un resultado de valor al usuario). Para cada CU describimos un conjunto de secuencias en donde cada una representa la interacción del sistema y sus abstracciones claves con los elementos externos del mismo [3]. Estos elementos son actores que se modelan y se definen sus interrelaciones en el Modelo de Actores del Sistema (MAS).
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. Surge también la importancia construir incrementalmente el MMU(Modelo de Mensajes de Usuario) en la medida que necesitamos dejar explicito una interacción del sistema con el usuario, desde escenarios del caso de uso.

4.3 Descripción de la elaboración del CIM en la etapa de Análisis OO

La etapa de Análisis se describe a partir de cuatro modelos fundamentales. El Modelo de Secuencia del Sistema (MSS) define, para un escenario particular de un caso de uso, los eventos que los actores generan, su orden y los eventos que se intercambian entre sistemas. El Modelo de Clases Análisis (MCA) se utiliza para identificar y visualizar las relaciones entre las clases conceptuales que involucran el sistema, relacionadas con el escenario que se está diseñando en base a objetos derivados de la etapa inicial, más precisamente del MOD y de las conclusiones del estudio del MCU.
El Modelo de Transición de Estados (MTE) enfatiza el comportamiento dependiente del tiempo del sistema. 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 la forma en que un elemento se puede mover entre estados, clasificando su comportamiento de acuerdo con los disparadores de transiciones y las guardas de restricciones.
Con el fin de interrelacionar todos los artefactos independientes por cada caso de uso descritos ut supra, se aplica el modelo de Realización de Casos de Uso en Análisis (RCUA), proporcionando una vista integradora de lo producido en esta etapa centrado en el caso de uso. De este modo podemos empezar a observar la evolución del caso de uso en 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 MDA, que será insumo para construir el PIM. El cual se comenzara a construir a continuación.

4.4 Descripción de la elaboración del PIM en la etapa de Diseño OO

Se describen tres disciplinas que expanden y especifican todo lo realizado en la etapa de Análisis. El objetivo de esta etapa es obtener un PIM preciso. El Modelo de Clases de Diseño (MCD) es la evolución del MCA. Contiene clases de software concretas e independientes de la plataforma de implementación, a las cuales se le agregan restricciones OCL, para eliminar ambigüedades y así aclarar su comportamiento, responsabilidades, estado interno e implementar reglas de dominio. El Modelo de Interacción (MINT) está formado por diagramas de interacción que describen cómo colaboran las clases de diseño, a través de los mensajes entre los objetos, en uno de los distintos escenarios de un caso de uso en particular. En el Diagrama de Realización de Casos de Uso en Diseño (RCUD) se relacionan todos los artefactos descritos en esta etapa (MCD y MINT) centrados en cada caso de uso de la misma forma que en las etapas de Requerimientos y de Análisis.

4.5 Mecanismos de trazabilidad con herramientas CASE en el Project Model

Existen dos funcionalidades muy importantes que brindan las herramientas CASE: la matriz de relaciones y la trazabilidad. Estas nos permiten hacer un seguimiento de la evolución de los artefactos del sistema, vinculando y generando un sustento a cada uno de los elementos de nuestro modelo, depurando relaciones innecesarias y observando, de manera sencilla, la dependencia entre objetos de modelado y el impacto que podría originar algún cambio en ellos.
La matriz de relaciones, en particular, nos muestra los vínculos existentes entre los elementos de un modelo de origen y un modelo de destino. Estos elementos deben estar relacionados al menos una vez, lo que significa que podemos ver la procedencia de los componentes que indiquemos.
Por ejemplo, como vemos en la Figura 7, el MBR debe darle soporte a los requerimientos diagramados en el MRF; los MBR bajo análisis definen los límites del modelado posterior.
La ventana de trazabilidad (Figura 8) permite ver cómo fue evolucionando el artefacto desde sus orígenes, el sustento que le brinda a otros elementos de modelado, los vínculos que se generan y conectan todas las vistas que proveen descripciones a distintos niveles de abstracción. Los modelos más abstractos son útiles para la descripción y documentación, mientras que los más depurados son ventajosos para poder comprender mejor las fases del ciclo de vida, siendo la trazabilidad una herramienta que nos permite transmitir los objetivos concretos desde etapas tempranas del desarrollo hasta su finalización.
En forma complementaria es necesario hacer diversas estimaciones, como una parte central de la administración de proyectos. Luego de realizar distintas comparaciones, podemos concluir que la técnica Use Case Points (UCP) es la que nos brinda valores muy cercanos a la realidad, dado que centramos todo nuestro modelado y desarrollo sobre los casos de uso (metodología RUP). Al utilizar la herramienta CASE, pudimos realizar de manera exitosa la evaluación de las distintas técnicas y obtención de los resultados.

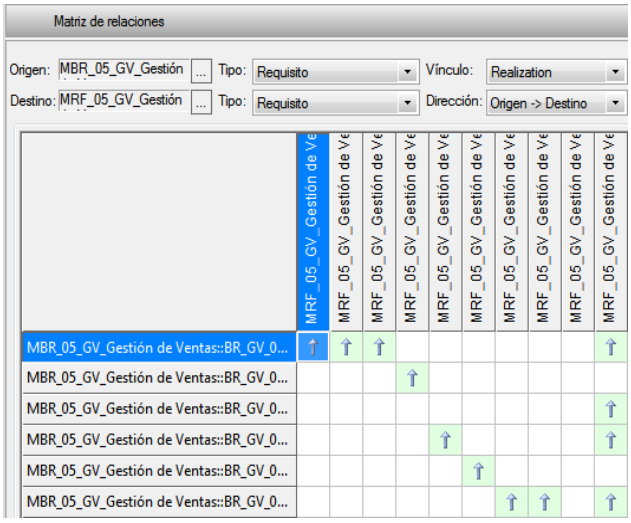


Fig. 7. Matriz de relaciones: explicitando conexión de elementos del MBR con el MRF

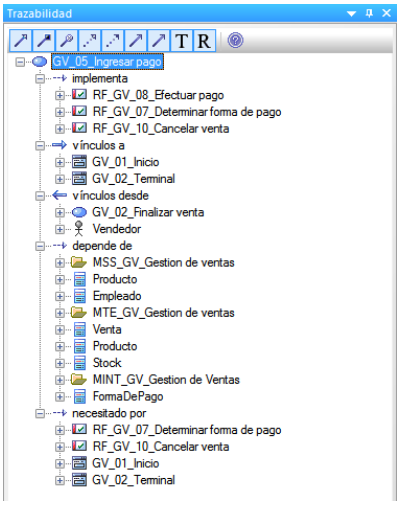


Fig. 8. Vista de Trazabilidad: jerarquía de conexión con elementos de otros modelos centrados en el CU seleccionado

5. Conclusiones y líneas de trabajo futuro

El desarrollo de un proyecto de modelado, siguiendo las buenas prácticas de la ingeniería de software y lineamientos RUP, permitió abstraernos de las tecnologías subyacentes haciendo énfasis en la construcción de modelos robustos que cubran los requisitos de negocio. El uso de herramientas CASE facilitó la navegabilidad y la trazabilidad de dichos modelos, favorenciendo la consistencia de los modelos

CIM/PIM construidos a partir de los lenguajes notacionales UML, SYSML y OCL que dan soporte a MDA.
Aunque el desarrollo realizado nos permite fijar las bases necesarias para la construcción - a través de transformación de modelos MDA - de los módulos de implementación y prueba – incluidos como parte del proyecto –, sólo fueron nombrados porque sobrepasan los límites de este trabajo. A futuro, trazaremos objetivos tales como definir perfiles de UML que permitan generar mecanismos para extender y adaptar metaclases de un metamodelo cualquiera – y no sólo el de UML – a las necesidades concretas de una plataforma – como puede ser .NET o J2EE – o de un dominio de aplicación – como tiempo real, modelado de procesos de negocio, etc. – [17][18], para realizar transformaciones de modelos PIM a modelos PSM.

6. Referencias

1. Pons Claudia, Giandini Roxana y Pérez, Gabriela. *Desarrollo de Software Dirigido por Modelos. Conceptos teóricos y su aplicación práctica* 1er. Edición. EDULP & McGraw- Hill Educación. Argentina (2010). ISBN-13: 978-950-34-0630-4
2. Object Management Group: http://www.omg.org
3. Object Management Group, MDA specification: http://www.omg.org/mda/specs.htm
4. UML 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
5. OMG SysML v. 1.2 Specifications: http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf
6. 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
7. lan Sommerville. *Ingeniería del Software*. Séptima edición. Pearson Education, S.A. (2005). ISBN: 84-7829-074-5
8. Roger Pressman. *Ingeniería del Software un enfoque práctico*. Sexta Edición. Mc Graw-Hill. (2005). ISBN-13:978-970-10-5473-4
9. Alistair Cockburn. *Agile Software development: The Cooperative Game*. 2nd Edition. Addison-Wesley Professional. (2007). ISBN-13: 978-032-14-8275-4
10. Richard F. Paige, Gøran K. Olsen, Dimitrios S. Kolovos, Steffen Zschaler and Christopher Power. Building Model-Driven Engineering Traceability Classifications. ECMDA. Traceability Workshop 2008.
11. Ismenia Galvao and Arda Goknil. Survey of Traceability Approaches in Model-Driven Engineering. 11th IEEE International Enterprise
Distributed Object Computing Conference 2007.
12. Jane Huffman Hayes. Grand Challenges in Traceability, Center of Excellence for Traceability. University of Kentucky 2009.
13.http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
14. Geert Monsieur, Monique Snoeck, Raf Haesen and Wilfried Lemahieu. PIM to PSM transformations for an event driven architecture in an educational tool. ECMDA. (2006).
15. Keeton, Marlys. *Microsoft Solutions Framework (MSF): A Pocket Guide*. Van Haren Publishing. (2006). ISBN-13: 978-907-72-1216-5.
16. http://www.sparxsystems.com.au/products/ea/index.html
17. Ejemplos de perfiles UML: www.uml.org.cn/UMLSearch/UMLExampleProfiles.pdf
18. Perfiles UML: www.uml-diagrams.org/profile-diagrams.html