Los procesos ágiles en la producción de productos software en ambiente MDD

Leopoldo Nahuel[1,2], **Matías Mangano**[1], **Lautaro Mendez**[1], **Roxana Giandini**[1,2]
{lnahuel, mmangano, lmendez, rgiandini}@linsi.edu.ar

[1] Laboratorio de Innovaciones en Sistemas de Información, LINSI. Universidad Tecnológica Nacional.
Facultad Regional La Plata. Calle 60 esq. 124. La Plata, Buenos Aires, Argentina
[2] Laboratorio de Investigación y Formación en Informática Avanzada, LIFIA. Universidad Nacional de La Plata.
Facultad de Informática. Calle 50 y 12. La Plata, Buenos Aires, Argentina

Resumen: *El Desarrollo de Software Dirigido por Modelos (MDD, Model Driven Development) es actualmente un importante paradigma en la Ingeniería de Software, proponiendo radicalmente sustituir — como artefacto principal en el proceso de construcción de productos software — al código fuente de lenguajes de programación por modelos. Éstos son considerados entidades primarias, creados desde etapas tempranas del proceso de desarrollo, permitiendo así nuevas posibilidades de crear, analizar y manipular sistemas complejos a través de lenguajes de modelado y herramientas que automatizan la transformación entre modelos, de mayor a menor nivel de abstracción. Este paradigma aun hace notar la falta de integración de procesos sólidos, que permitan agilizar la producción y el mantenimiento de productos software. En este contexto, este paper presenta la aplicación de los beneficios y mejores prácticas que promueven los modernos procesos ágiles de desarrollo en un ámbito de producción de software que sigue lineamientos del paradigma MDD, destacando ventajas y problemas que éste resuelve durante la concepción y elaboración de productos software.*

Abstract: *The Model Driven Development (MDD) is actually an important paradigm in Software engineering, proposing to radically replace –as main artifact in the process of software product construction – the source code in the driven by model language. These are considered primary entities, created in early stages allowing new possibilities of creating, analyzing and manipulating complex systems throughout model language and tools that automates transformations between models. This paradigm still makes notice the lack of integration of solid processes that allow the maintenance and software agile production. In this context, we shall present the application of benefits and better practices that promote modern and agile processes of development in the sphere of software production following guidelines of the MMD paradigm, highlighting benefits and problems it solves during the elaboration of software products.*

Palabras clave: Producción de Software, Proceso Ágil, Desarrollo Dirigido por Modelos, Transformación de Modelos.

1. Introducción

El software, conjunto finito y definido de acciones primitivas ejecutadas por una computadora que tiene por objeto la ejecución de tareas específicas, posee una característica que lo distingue del resto de los productos de la industria: su intangibilidad. Aun así resulta innegable su utilidad al momento de modelar aspectos de la realidad, automatizar y agilizar tareas de cálculo y simular procesos de diversos tipos. Puede verse claramente que el software no solo constituye un producto en sí mismo, sino también una valiosa herramienta para la producción en áreas muy diversas.

Se ha evidenciado una evolución en la producción y mantenimiento de Productos Software, a partir del uso de modelos como artefactos principales de desarrollo, proponiendo un incremento en los niveles de abstracción del cual resulte una construcción más rápida, eficiente, reutilizable, independiente de la plataforma tecnológica y flexible a la modificación de los requerimientos iniciales para los cuales el producto software fue construido [Acuña2005].

Para lograr que dichos productos tengan estas características tan deseables y fructíferas en cualquier ámbito productivo de la industria actual, nos serviremos del uso de un emergente paradigma del campo de la Ingeniería de Software: el Desarrollo Dirigido por Modelos (bien conocido en el ámbito de la industria de software como MDD, por sus siglas en inglés) [Pons2010] y del interesante aporte de las metodologías ágiles para la producción de productos software [Calderón 2007].

El resto de este paper está organizado de la siguiente manera. En la Sección 2, se presenta el problema actual en fases de producción y mantenimiento de productos software. La Sección 3 ofrece una visión general del marco de trabajo en MDD (modelos, herramientas, lenguajes), sus beneficios y su proceso de desarrollo. En la Sección 4, se muestra el aporte de las metodologías ágiles modernas aplicables en distintos ámbitos de producción de software. La evaluación de aplicabilidad de procesos ágiles en ámbitos de producción MDD se encuentra esquematizada en la Sección 5, y finalmente, las conclusiones finales y líneas de trabajo futuro está en la Sección 6.

2. Contexto del problema

Se expondrán aquí las dificultades que se evidencian a la hora de producir y mantener productos software. Las causas de estos inconvenientes tienen como factores dominantes la complejidad propia de la tarea de construir un software y los numerosos cambios que éste debe sufrir con fines de adaptarse a las necesidades cambiantes de los usuarios y a las nuevas tecnologías. Si bien estos problemas, los cuales dieron origen al concepto de "crisis del software", datan de la década del sesenta; siguen vigentes en la actualidad, debido a la creciente

complejidad de los sistemas a desarrollar y de la fiabilidad que se requiere de éstos en diversas ocasiones.

A continuación haremos una breve reseña de la evolución de estos problemas, de las diferentes soluciones con las cuales se intentó enfrentarlos y del creciente protagonismo que los modelos han desempeñado a través del tiempo.

En un principio, la construcción de un software se focalizaba fuertemente en la escritura de código fuente compilable, asegurando que el programa resultante funcione cumpliendo los objetivos esperados. Este método imposibilitaba la buena interacción y coordinación en grandes grupos de trabajo para el desarrollo de software complejo y robusto, ya que el código fuente resulta de difícil comprensión para quien no lo ha escrito, haciendo contraproducente la inclusión o reemplazo de programadores. Resultando, además, de imposible lectura para aquellos usuarios que no están familiarizados con la informática, lo cual dificultaba aún más la participación activa multidisciplinaria durante el proceso de desarrollo del software.

A finales de la década de los 70, se planteó la idea de que un proyecto de software debía ser encarado de la misma manera que cualquier otro propósito ingenieril, y para eso se tomó la idea de "modelo" como parte fundamental del proceso de construcción de sistemas con alto contenido de software y se definió un cuerpo de conocimientos englobado bajo el nombre de Ingeniería de Software Basada en Modelos (conocido como MBE, por sus siglas en inglés). Siguiendo una metodología de desarrollo basado en modelos, se debe comenzar con la elaboración de un modelo que conceptualice de manera abstracta al problema, permitiendo un análisis que posteriormente de lugar a un plan de implementación que contemple todo lo aprendido del estudio de ese modelo. Para construir estos modelos se utilizan lenguajes gráficos estándar como UML [Booch2006]. Si bien MBD solucionaba, de manera eficaz, algunos problemas en el desarrollo de sistemas software, daba lugar a otros, como ser: dificultad de mantenimiento, problemas de productividad, documentación y falta de flexibilidad a los cambios tecnológicos [Ambler2002].

Actualmente, los modelos se consideran componentes más activos dentro del ciclo de vida de un proyecto de software y se dejó atrás la idea de modelos estáticos que servían en primera instancia, pero al comenzar a codificar quedaban desactualizados casi inmediatamente y por ende obsoletos. Este nuevo paradigma se denominó internacionalmente como "Modeling Driven Development" (MDD). La visión del "Desarrollo de software Dirigido por Modelos" posiciona a los modelos como entidades primarias y productivas que guían al desarrollo de un sistema, en forma activa durante todo el proceso de elaboración y construcción de productos software. Actualmente, MDD se encuentra implementado bajo el estándar internacional "Model Development Architecture" [MDA2003] regulado por el consorcio "Object Management Group" [OMG2012]. MDA es un marco de trabajo tecnológico para el desarrollo de productos software que permite organizar y estructurar sistémicamente arquitecturas empresariales, soportado por potentes herramientas tipo CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) que automatizan, entre otros, la edición de modelos y facilitan las transformaciones entre los mismos.

3. Proceso para Producción de productos software en ambiente dirigido por modelos

El desarrollo dirigido por modelos nace como idea innovadora para dar solución a los problemas clásicos que se presentan en la Ingeniería de software. La idea principal que subyace a este paradigma es la generación automática de código fuente a través de transformaciones automatizadas aplicadas a modelos completos y consistentes. De esta manera se procede con el modelado desde el nivel más abstracto hasta el más concreto, definiendo distintos modelos para cada nivel de abstracción. En la figura 1, se muestra, en forma más ilustrativa, la cadena de modelos y transformaciones del proceso iterativo e incremental [Larman2004] en el contexto del ciclo de vida MDD.

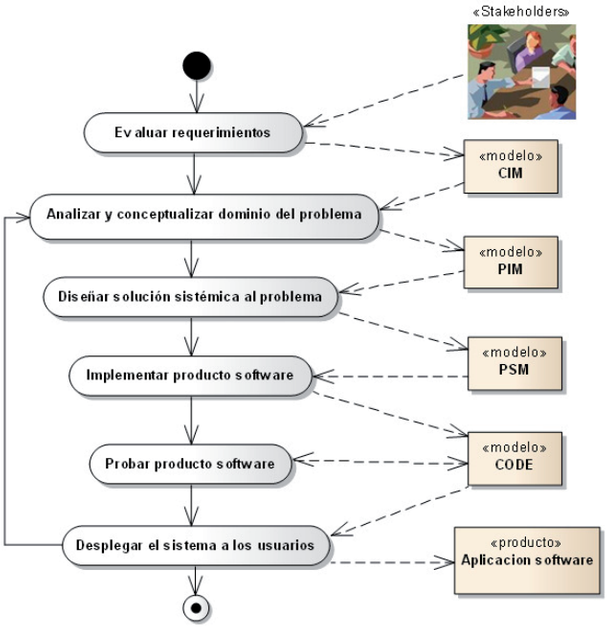

Figura 1. Proceso iterativo y elaboración de producto software para el ciclo de vida MDD.

El proceso MDD distingue 4 tipos de modelos bien conocidos por sus siglas en inglés:

CIM (Computational Independent Model): describen la lógica del dominio del negocio desde una perspectiva independiente de la computación. Estos modelos están destinados a usuarios que no poseen conocimientos técnicos acerca de los artefactos que se utilizarán para la implementación ya que no muestra detalles de la estructura del sistema.

PIM (Platform Independent Model): describen de forma abstracta el dominio y funcionalidad del sistema de forma independiente a cualquier tecnología de implementación, como ser: sistemas operativos, lenguajes de programación, hardware, etc.

PSM (Platform Specific Model): describen el sistema en términos de construcciones implementativas específicas ligadas a una tecnología concreta. A partir de un PIM, se procede a la generación automática de uno o más PSM los cuales son la proyección del PIM en una plataforma específica. De esta manera podemos obtener múltiples PSM para diferentes tecnologías de implementación, como ser: java, C++, C#, Python, php, etc.

CODE o IM (Implementation Model): describe o especifica el sistema en término del código fuente (modelo texto) en una tecnología concreta. Como etapa final en este ciclo de vida MDD, se procede a transformar cada PSM a código fuente compilable.

Como se detalló, los modelos se suceden desde el más abstracto hasta el más concreto. En principio, se procede en la elaboración del CIM y a partir de éste se crea el PIM. Obviamente no es posible realizar esto automáticamente ya que es imposible determinar qué requisitos de negocio deben ser implementados y de qué manera hacerlo. En la Figura 2, vemos ilustrado el proceso de transformaciones entre modelos.

El punto clave de MDD es la posibilidad de generar, a partir del PIM, el PSM mediante transformaciones automáticas implementadas por herramientas creadas para tal fin y a partir de uno o varios PSM obtener, también mediante transformaciones, el código fuente de la aplicación.

Es importante destacar la posibilidad de generar desde un mismo PIM, PSM orientados a distintas plataformas de implementación, a su vez esto nos permite generar código fuente para distintas tecnologías con poco esfuerzo gracias a las transformaciones automáticas, como se ilustra en la figura 3.

Dado que el PSM y el código se generan mediante computadoras automáticamente, los modelos previos de los cuales se parte deben estar ausentes de ambigüedades y ser consistentes en su construcción. A grandes rasgos podemos ver a una transformación como una caja cerrada que tiene por entrada un modelo fuente, PIM o PSM, y por salida un modelo resultado, PSM o Código respectivamente. En general el modelo del cual se parte tiene un nivel de abstracción mayor que el resultante de la transformación.

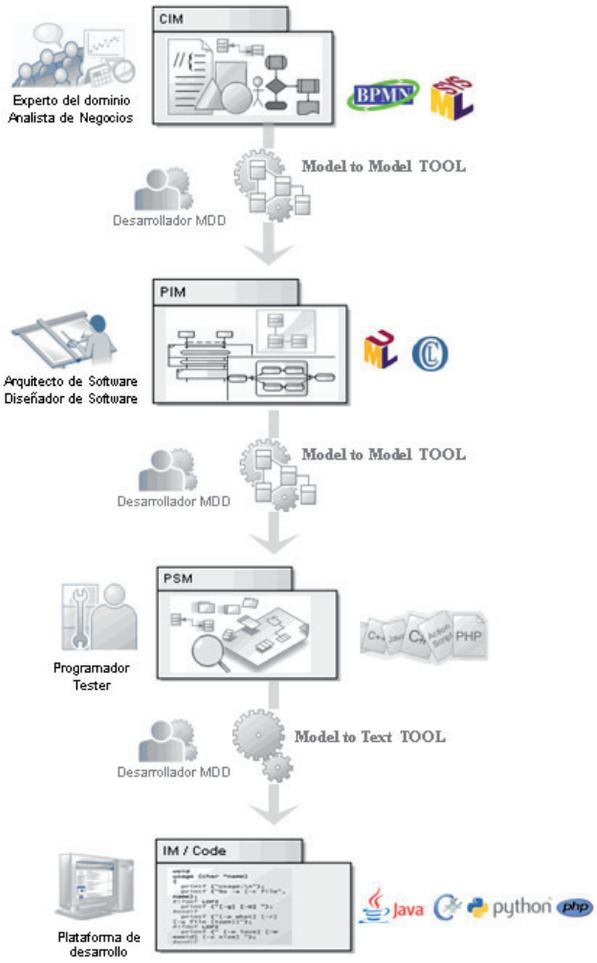

Figura 2. Evolución y transformación de modelos en proceso de desarrollo MDD: roles, modelos y tolos.

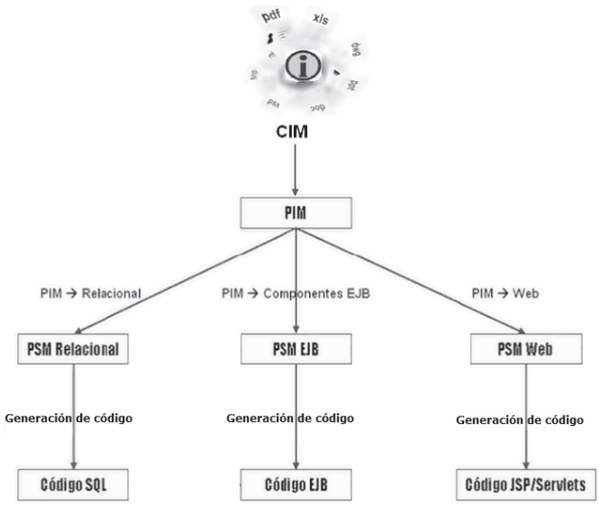

Figura 3. Ciclo de vida MDD aplicado a varios PSM y su transformación a código (modelo de implementación).

Dada la importancia del modelado, en MDD es necesaria la utilización de lenguajes de modelado sustentados por una definición formal que permita la construcción de modelos precisos, comprensibles, consistentes y completos. En la actualidad, el lenguaje gráfico para modelado de mayor uso es el UML (Unified Modeling

Language) el cual posee una sintaxis gráfica amigable. La base formal necesaria para la aplicación de UML sobre MDD fue provista por OMG a partir de Meta Object Faciliy (MOF) [MOF2011]. Para agregar restricciones y proveer de mayor precisión a modelos escritos en lenguajes gráficos, el OMG estandarizó un lenguaje textual basado en lógica de primer orden llamado Object Constraint Language (OCL) [Warmer2003].

En lo que se refiere a la construcción y control de los diversos modelos, será fundamental el uso de aplicaciones, llamadas herramientas CASE (Computer Aided Software Engineering) [Sommerville2005], que proveen utilidades para la edición de modelos y su documentación, análisis de impactos frente a cambios, estimación de costes, versionado, navegación de un modelo a otro, entre otras. En particular, cuando se modela, estas herramientas CASE proporcionan algunas funcionalidades que resultan muy beneficiosas en el ámbito MDD, específicamente nos referimos a la trazabilidad (la cual permite visualizar la evolución de los artefactos del modelo, así como también las dependencias e impacto de cada uno de ellos) y a la matriz de relaciones (funcionalidad que expone los vínculos existentes entre un modelo origen y un modelo destino).

3.1. Planificación y administración de proyectos sistémicos basado en línea de producción MDD

Puede establecerse una subdivisión del proyecto, en interno y externo. En el primero, una parte del equipo produce las herramientas MDD necesarias para el desarrollo subsiguiente; mientras que en el segundo, el resto de los desarrolladores se encarga de crear la aplicación utilizando las herramientas previamente construidas por el equipo interno.

Este acoplamiento supone una exigencia adicional al momento de organizar y planificar, puesto que aumentan las dependencias entre los desarrolladores. Pero, por otro lado, permite distribuir el esfuerzo entre ambos proyectos según se crea conveniente.

La delimitación entre ambos proyectos no es de carácter estático, por el contrario, algunos miembros participan en ambas partes del proyecto. Sin embargo, es importante aclarar que los desarrolladores más experimentados pueden especificar con mayor precisión la automatización del proceso de desarrollo.

Si bien la planificación y el seguimiento del proceso de desarrollo son muy similares al de cualquier proyecto, resulta recomendable separar el desarrollo en varias iteraciones de tiempo. La importancia de esto será tratada con un mayor grado de detalle en las secciones posteriores.

Acerca del seguimiento es aconsejable tener en cuenta algunos aspectos relevantes, a saber: el proyecto no solo generará código, sino también documentos, configuraciones, reportes y casos de prueba; será conveniente asegurar que el proceso soporta ambientes de prueba y se deberá considerar si las herramientas MDD serán reusables en proyectos futuros.

3.2. Reuso de artefactos durante el proceso de construcción: beneficios

Como en cualquier otro proceso productivo, el ahorro o el reúso de ciertos elementos es una propiedad muy deseada, ya que nos permite establecer estándares de construcción, como así también reducir costes y tiempo de desarrollo. En MDD, los artefactos de mayor importancia son modelos, transformaciones y el código resultante a través del proceso de transformación de modelos. Particularmente, el reúso de los modelos y las transformaciones tienen mucho valor, ya que en ellos se encuentran incorporados el conocimiento y la experiencia de los especialistas que estuvieron a cargo de su construcción, quedando ésta disponible para su uso aun después de que estas personas dejen de formar parte del proyecto [Hazzan2008].

Dadas las características inherentes a un proyecto dirigido por modelos, el reúso es una propiedad que no debe desaprovecharse. En los modelos de alto nivel de MDD, resulta más fácil determinar puntos en común entre lo que se necesita construir y lo que fue construido.

4. Enfoques de proceso liviano para producción de productos software: Metodologías ágiles

Las metodologías tradicionales de la Ingeniería de Software se enfocan en el riguroso cumplimiento de un plan de proyecto, que ha sido definido en la fase inicial del desarrollo. Conceptualizan a la creación de software como un proceso meticulosamente definido, con pasos bien delimitados, de requerimientos estáticos y exhaustivamente documentado. Esto ocasiona que el proceso carezca de flexibilidad ante cambios en los requisitos iniciales. La participación del usuario se limita a la primera etapa del proceso y por ende la realimentación con los diseñadores e implementadores de software es muy bajo o nulo.

En contraposición a esta visión, surgen metodologías iterativas, dinámicas y adaptativas llamadas Metodologías Ágiles, centradas fuertemente en la relación con los usuarios. Las ideas y conceptos que dan fundamento a este grupo de métodos se encuentran reunidos en un documento escrito en la década de los noventa denominado “Manifiesto Ágil”. Éste promueve la colaboración de los usuarios en todo el proceso de desarrollo; respuesta ante el cambio, lo cual permite que los requisitos se modifiquen incluso en etapas avanzadas del desarrollo; entrega temprana y continua de software operativo; disminución en el intercambio engorroso de documentación a partir de la comunicación cara a cara entre los miembros y centrado en los individuos que forman parte del proyecto y su interacción.

La cualidad de iterativas supone la división del proyecto en períodos (iteraciones) en los cuales se llevan a cabo tareas de planificación, análisis de requisitos, diseño, codificación, prueba y documentación; pretendiendo obtener un software funcional luego de cada iteración.

A lo largo de los años, se dieron a conocer diversas metodologías ágiles, conocidas mayormente por sus siglas

en inglés, como: XP (eXtreme Programming), AM (Agile Modeling), TDD (Test Driven Development), AUP (Agile Unified Process), DSDM (Dynamic Systems Development Method), SCRUM, Cristal Orange, entre otras.

5. Procesos ágiles en ambientes de producción basado en líneas de trabajo MDD

Se han expuesto las obvias ventajas de la aplicación de una filosofía ágil para el desarrollo de software y por ende es importante adaptar estos conceptos al paradigma MDD. Una de las ideas principales adoptadas por la mayoría de las metodologías ágiles es la división del proyecto en breves iteraciones de tiempo. Esto nos permite disminuir la incertidumbre manteniendo rumbos claros y objetivos precisos. En una primera iteración, se generan una versión inicial de los modelos de alto nivel y sus transformaciones.

Esta práctica es beneficiosa ya que nos provee las siguientes ventajas:

- Puede obtenerse una participación mucho más activa y crítica por parte de los usuarios.
- Se alcanzan resultados rápidamente con un esfuerzo mínimo, de esta manera se reduce el escepticismo y se aumenta la confianza en el grupo.
- Al generar rápidamente artefactos funcionales, el equipo de desarrollo gana experiencia y se hace posible una estimación del tiempo y esfuerzo que tomará el resto del proyecto.
- Se aumenta la productividad, ya que se mantiene ocupado a todo el equipo, de esta manera no es necesario que los implementadores de transformaciones esperen a la finalización de un PIM completamente definido por parte de los modeladores.
- En etapas posteriores se trabaja sobre lo aprendido previamente contando con una base sólida gracias a la experiencia ganada.

Los conceptos del paradigma "desarrollo de software dirigido por modelos" son muy recientes y todavía no han sido adaptados completamente por el mundo informático, menos aún por la industria. Por esta razón es que la mayoría de las metodologías de desarrollo de software (tanto las tradicionales como las ágiles) no han sido pensadas para la aplicación directa sobre MDD. Si tenemos en cuenta el cambio radical en el enfoque dirigido por modelos, donde el modelado y sus artefactos son de vital importancia, conducen el desarrollo completo y la generación automática de código a partir de transformaciones aplicadas a los modelos, es razonable preguntarse si la aplicación directa de una metodología que no fue pensada para soportar este nuevo paradigma es viable.

En suma, existen algunos aspectos en los cuales los principios ágiles y MDD se contraponen:

- Importancia de las herramientas de desarrollo: en MDD resultan fundamentales, mientras que en el desarrollo ágil no es así.
- Énfasis en el modelado: si bien es el aspecto fundamental del paradigma de desarrollo dirigido por modelos, consta de una importancia marginal en los procesos ágiles. En estos últimos, los esfuerzos están dirigidos al testeo y a la codificación.

Pero, por otra parte, algunas desventajas de las metodologías ágiles serían subsanadas por el paradigma MDD, y viceversa. El uso de metodologías ágiles permite cambios sucesivos en los requerimientos, pudiendo ser necesaria una reescritura de código en diversas áreas del producto software [Stober2009]; las transformaciones de MDD solucionarían esto de un modo muy eficiente, pues solo sería necesario realizar las modificaciones en el modelo, y éstas se traducirían al código en forma automática.

También se debe tener en cuenta que en un principio el MDD proponía una fase de análisis y modelado exhaustivas, antes de continuar con otras actividades del desarrollo, claramente, este enfoque no se muestra muy flexible a cambios en los requerimientos y limita la participación del usuario a las primeras fases de desarrollo, lo cual puede provocar incertidumbre en el usuario que desea ver parte del producto. Esta dificultad puede mitigarse fusionando el MDD con las propuestas ágiles, lo cual supondría desarrollar modelos de manera rápida desde las primeras iteraciones, privilegiar la participación activa del usuario y fomentar la comunicación entre los individuos bajo una filosofía ágil de desarrollo de software en un contexto MDD.

Scott W. Ambler ha impulsado y promovido conceptos propios de las metodologías ágiles para la construcción de software centrándose específicamente en el modelado. Ambler propuso una innovadora versión de MDD, denominada Agile Model Driven Development (AMDD) [Ambler2004], la cual él mismo define como la versión ágil de MDD. Mientras que MDD plantea la construcción exhaustiva y completa de modelos antes de escribir código, AMDD propone la construcción de modelos que sean "lo suficientemente buenos" como para dirigir y continuar con el desarrollo, para luego refinar en iteraciones posteriores.

5.1. Análisis de resultados y aportes

La evaluación realizada en la sección anterior ofrece una visión global acerca de la integración de procesos ágiles dentro del ciclo de vida del paradigma MDD, y al mismo tiempo aporta valor al marco metodológico definido en la iniciativa del proyecto de investigación y desarrollo "Modelado ágil para la producción de software en MDD aplicable a Gestión de Procesos de Negocio (BPM)" [Giandini2011].

Este proyecto de I&D se encuentra actualmente adaptando un prototipo de herramienta CASE (implementado con tecnología open source Eclipse), de soporte a la edición y transformación de modelos CIM y PIM en un contexto de modelado ágil para la producción de software en MDD, tomando en consideración el

estudio de aspectos de modelado en los procesos ágiles modernos y realizando pruebas conceptuales de su aplicabilidad en etapas tempranas del proceso de modelado.

Esta evaluación sobre aplicabilidad de enfoques ágiles está actualmente siendo considerada en los alcances de la implementación de la herramienta CASE, haciendo fuertemente hincapié en el modelado de Procesos de Negocio utilizando el lenguaje gráfico BPMN (Business Process Management Notation) [BPMN2011] como estrategia ágil para vislumbrar los requerimientos de negocio (porciones significativas de modelos CIM – vista de negocio) y ofrecer su rápida transformación en porciones de modelos sistémicos (porciones significativas de modelos PIM – vista de sistema).

6. Conclusiones y trabajos futuros

Debido a la importancia que el software supone en el escenario productivo actual y a sus diversas aplicaciones en la industria queda en evidencia que no solo constituye un producto en si mismo sino también que requiere la creación y aplicación de procesos formales para su construcción.

El paradigma MDD resulta en un aporte sustancial para la producción de software, pero aún no existe una metodología que lo sustente por completo. Tanto las metodologías tradicionales, como las ágiles muestran algunos aspectos negativos para este incipiente paradigma.

De todos modos podemos inferir que pequeños cambios adaptativos en los procesos ágiles serían la clave para lograr el eclecticismo buscado entre este paradigma y las metodologías de desarrollo. El proceso AMDD parece ser el que mejor ha podido aunar las características positivas de MDD con el enfoque ágil aplicable a la producción de productos software, siendo una alternativa muy atractiva debe ser examinada y perfeccionada en lo sucesivo.

Como línea de trabajo futuro, se espera madurar un esquema de integración ágil que incluya buenas prácticas de modelado BPMN y tecnologías de transformación de modelos, de apoyo al trabajo en el ámbito BPM. Incorporar la concepción de modelos CIM escritos en lenguajes BPMN y SysML (System Modeling Language) [SysML2012] combinados, es uno de los retos importantes como continuación del presente trabajo.

Referencias bibliográficas

[Acuña2005] Acuña, S. y Juristo, N. “Software Process Modeling”. Springer Science-Business Media, Inc., pp. 26-36 (2005). International Series in Software Engineering, Vol. 10. ISBN-10: 0387242619

[Ambler2002] Ambler, S. “Agile modeling: effective practices for eXtreme programming and the unified process”. John Wiley & Sons, Inc., New York. Wiley Computer Publishing, pp. 20-26, 223-242 (2002). ISBN-10: 0471202827

[Ambler2004] Ambler, S. “The object primer: agile modelling-driven development with UML 2.0”. 3° Edición. Cambridge University Press, pp. 101-133 (2004). ISBN-10: 0521540186

[BPMN2011] OMG. Lenguaje notacional gráfico para BPM “Business Process Management Notation”. Versión 2.0 - http://www.omg.org/spec/BPMN/2.0/

[Booch2006] Booch, G., Jacobson, I. y Rumbaugh, J. “El lenguaje unificado de modelado”.2° Edición. Pearson Education S.A., pp. 3-40 (2006). ISBN-13: 9788478290765.

[Calderón2007] Calderón, A., Dámaris, S., Valverde Rebaza, J. “Metodologías Ágiles”, Trujillo, Perú (2007). http://seccperu.org/files/Metodologias%20Agiles.pdf.

[Giandini2011] Giandini, R. Director del Proyecto de Investigación & Desarrollo “Modelado ágil para producción de software en MDD aplicado a Gestión de Procesos de Negocio (BPM). PID homologado por Programa de Incentivos a los Docentes Investigadores del Ministerio de Educación de la Nación Argentina (código 25/I046) y por la SCTyP de la Universidad Tecnológica Nacional, Rectorado (código UTI1302).

[Hazzan2008] Hazzan, O. y Dubinsky, Y. “Agile Software Engineering”. Springer-Verlag London Limited, pp. 4-7, 8, 61-67 (2008). ISBN-10: 1848001983

[Larman2004] Larman, C. “Agile and Iterative Development: a manager's guide”. Pearson Education, Inc. Addison-Wesley Professional, pp. 9-20, 25-35 (2004). ISBN-10: 0131111558.

[MDA2003] OMG Contact: Watson, A. “Model Driven Architecture Guide”. Version 1.0.1, pp. 11-19 (2003). http://www.omg.org/cgi-bin/doc?omg/03-06-01.

[MOF2011] “Meta Object Facility (MOF) Core Specification”. Version 2.4.1, pp. 17-24 (2011). Release date: August 2011. http://www.omg.org/spec/MOF/2.4.1/PDF

[OMG2012] Object Management Group, OMG (2012). www.omg.org

[Pons2010] Pons, C., Giandini, R. y Pérez, G. “Desarrollo de Software Dirigido por Modelos: conceptos teóricos y su aplicación práctica”. 1°Edición. EDULP & McGraw-Hill Education, Argentina, pp. 28-38 (2010). ISBN-13: 9789503406304.

[Sommerville2005] Sommerville, I. “Ingeniería del Software”. 7° Edición. Pearson Education S.A., pp. 11, 79-83 (2005). ISBN-10: 8478290745.

[Stober2009] Stober, T. y Hansmann, U. “Agile Software Development: best practices for large software development projects”. Springer-Verlag Berlín Heidelberg, pp. 37-57 (2009). ISBN-10: 3540708308.

[SysML2012] OMG. Lenguaje notacional gráfico para Ingeniería de Sistemas “System Modeling Language”. Versión 1.3 - http://www.omg.org/spec/SysML/1.3/

[Warmer2003] Warmer, J. y Kleppe, A. “The Object Constraint Language: getting Your Models Ready for MDA”. Pearson Education, pp. 3-18 (2003), ISBN-10: 0321179366.