Evaluación y Aplicación de Procesos Ágiles para la Producción de Software en Ambientes de Desarrollo Dirigido por Modelos

R. Giandini[1,2*], L. Nahuel[1,2], N. Robles[2], M. Losada[2], M. Mangano[2], L. Mendez[2], I. Conte[2], M. Pérsico[2], I. Martínez[2], J. Perelli[2], N. Santos[2], P. Girado[2], L. Vargas[2], R. Di Girolamo[2]

1: Laboratorio de Investigación y Formación en Informática Avanzada – LIFIA. Facultad de Informática - Universidad Nacional de La Plata. Calle 50 y 120 – CP 1900 – La Plata – Buenos Aires
2: Laboratorio de Innovaciones en Sistemas de Información – LINSI. Departamento de Sistemas - Facultad Regional La Plata. Universidad Tecnológica Nacional. Av.60 esq. 124 s/n – CP 1900 – La Plata – Buenos Aires

*e-mail: rgiandini@linsi.edu.ar

Resumen. *El uso de modelos para construir distintos tipos de sistemas software es actualmente una de las claves para la producción de nuevas tecnologías. El Desarrollo de Software Dirigido por Modelos conocido por sus siglas en inglés "MDD" (Model Driven Development) se ha convertido actualmente en un importante paradigma de la Ingeniería de Software, proponiendo sustituir - como artefacto principal en el proceso de producción del software - al código fuente de lenguajes de programación por modelos. De este modo, tales modelos son considerados como entidades de primera línea, permitiendo nuevas posibilidades de crear, analizar y manipular grandes sistemas a través de diversos lenguajes de modelado y herramientas automáticas. En este ámbito, los aspectos de evolución y trazabilidad son un importante desafío teórico-práctico, necesarios tanto en actividades de modelado manual como en procesos de transformación automática entre modelos (que van desde el refinamiento de un modelo de negocio hasta llegar al código fuente compilable en una plataforma de implementación concreta). El motor productivo del MDD es utilizar herramientas automáticas dedicadas y establecer mecanismos de transformación estrictos para los distintos modelos (que van de los más abstractos a los más específicos) involucrados en el proceso de producción de software: CIM (Computational Independent Model), PIM (Platform Independment Model), PSM (Platform Specific Model) e IM (Implementation Model). En este trabajo presentaremos los resultados obtenidos sobre el estudio comparativo y aplicación de procesos ágiles en el campo sistémico aplicable al enfoque automatizado MDD para la producción de software, destacando aspectos evolutivos de productos intermedios durante el curso de transformación de los modelos hasta llegar al producto software resultante, subrayando la importancia del uso combinado de lenguajes de modelado y el apoyo de potentes herramientas CASE (Computer Aided Software Engineering) de soporte a la edición y transformación automatizada de modelos.*

Palabras clave: *Producción de software, Procesos ágiles, Desarrollo Dirigido por Modelos, Transformación automatizada de Modelos, Herramientas CASE, Lenguajes de Modelado.*

INTRODUCCIÓN

El producto software, conjunto finito y definido de acciones primitivas ejecutadas por una computadora que tiene por objeto la realización de tareas específicas, posee una característica que lo distingue del resto de los productos de la industria: su intangibilidad. Aún así resulta innegable su utilidad al momento de modelizar 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 si mismo sino también una valiosa herramienta para la producción en áreas muy diversas.

Se tratará de exponer aquí la 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 contrucción 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.

Para lograr que los productos de software tengan estas características tan deseables y fructíferas en cualquier ámbito productivo nos serviremos del uso de un emergente paradigma del campo de la Ingeniería de Software, el Model Driven Development (MDD) (Pons, et al., 2010) y del interesante aporte de las metodologías ágiles para la producción de software (Sommerville, 2005).

PROBLEMÁTICA

Se expondrán aquí las dificultades que se evidencian a la hora de producir un software. Las causas de estos inconvenientes tienen como factores dominantes la complejidad propia en las actividades necesaria para construir un software y los numerosos cambios que éste debe sufrir a fines de adaptarse a las necesidades cambiantes de los usuarios, como así también a las nuevas tecnologías del mercado. Si bien estas problemáticas, las 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, hasta concluir en el estado actual de la Ingeniería de Software en el contexto de la problemática planteada.

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 básicamente con los objetivos esperados por parte del usuario. Este método imposibilitaba la interacción y coordinación en grandes grupos de trabajo para el desarrollo de productos software de alta complejidad, ya que el código fuente resulta de dificil 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 tecnología informática de construcción del software, lo cual dificultaba aún más la participación activa y multidisciplinaria durante el proceso de construcción del producto software.

A finales de la década de los 70 se planteó la idea de que los proyectos sistémicos con importante cantidad de software debían ser encarados y encaminados de modo semejante que cualquier otro proyecto 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 Model Based software Engineering (MBE) (Sommerville, 2005). Siguiendo una metodología de desarrollo basado en modelos (MBD por sus siglas en ingles) como parte de MBE, se debe comenzar con el desarrollo de un modelo que conceptualice de manera abstracta al problema, permitiendo un análisis que posteriormente de lugar a un plan de construcción que contemple todo lo aprendido del estudio de ese modelo. Para construir estos modelos se utilizan lenguajes de modelado gráficos como UML (Grady y Rumbaugh, 2006). Si bien MBD solucionaba de manera eficaz algunos problemas en la construcción de sistemas software, daba lugar a otros, como ser: dificultad de mantenimiento, problemas de productividad, documentación y falta de flexibilidad a los cambios tecnológicos emergentes.

Actualmente los modelos comenzaron a considerarse como 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 implementar físicamente con lenguajes de programación específicos quedaban desactualizados y por ende obsoletos casi inmediatamente posterior a obtener una primera versión del producto software resultante. Este nuevo paradigma se denominó Modeling Driven Development (MDD) posicionando a los modelos como entidades primarias y fuertemente productivas, que guían al desarrollo de un sistema de forma activa durante todo el proceso de producción. Actualmente MDD es apoyado por la industria a través del estándar Model Development Architecture (MDA), (2012) regulado por el consorcio Object Management Group (OMG), (2012). En líneas generales, MDA ofrece una combinación de estándares tecnológicos (técnicas, herramientas, lenguajes de modelado y metamodelado, entre otros) como marco de trabajo para la producción de productos de software comercial siguiendo lineamientos definidos por la visión MDD, soportado por herramientas dedicadas para la construcción de modelos precisos y consistentes, favoreciendo la ejecución de motores para transformación automatizada entre diversos modelos (con distintos niveles de abstracción) hasta llegar al producto software resultante.

PROCESO PARA PRODUCCIÓN DE PRODUCTOS SOFTWARE EN MDD

El desarrollo dirigido por modelos nace como una idea innovadora para dar solución a los problemas clásicos que se presentan en la Ingeniería de Software. La idea principal que subyase a este paradigma es la generación automática de código 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 evolución de los distintos modelos a través del flujo de actividades del proceso MDD, que van desde la evaluación de requerimientos del negocio hasta el despliegue del producto software a los usuarios, para su posterior validación y aceptación. En caso de requerir ajustes y/o mejoras funcionales acerca del producto final (aplicación software), el proceso prevé comenzar el ciclo de refinamiento de modelos – partiendo desde el PIM – realizando los ajustes de funcionalidad o nuevos requerimientos sobre la solución lógica del dominio, así luego continuar con el resto de las actividades del proceso y transformaciones de modelos, para volver a entregar una nueva versión a los usuarios.

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 Independment 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

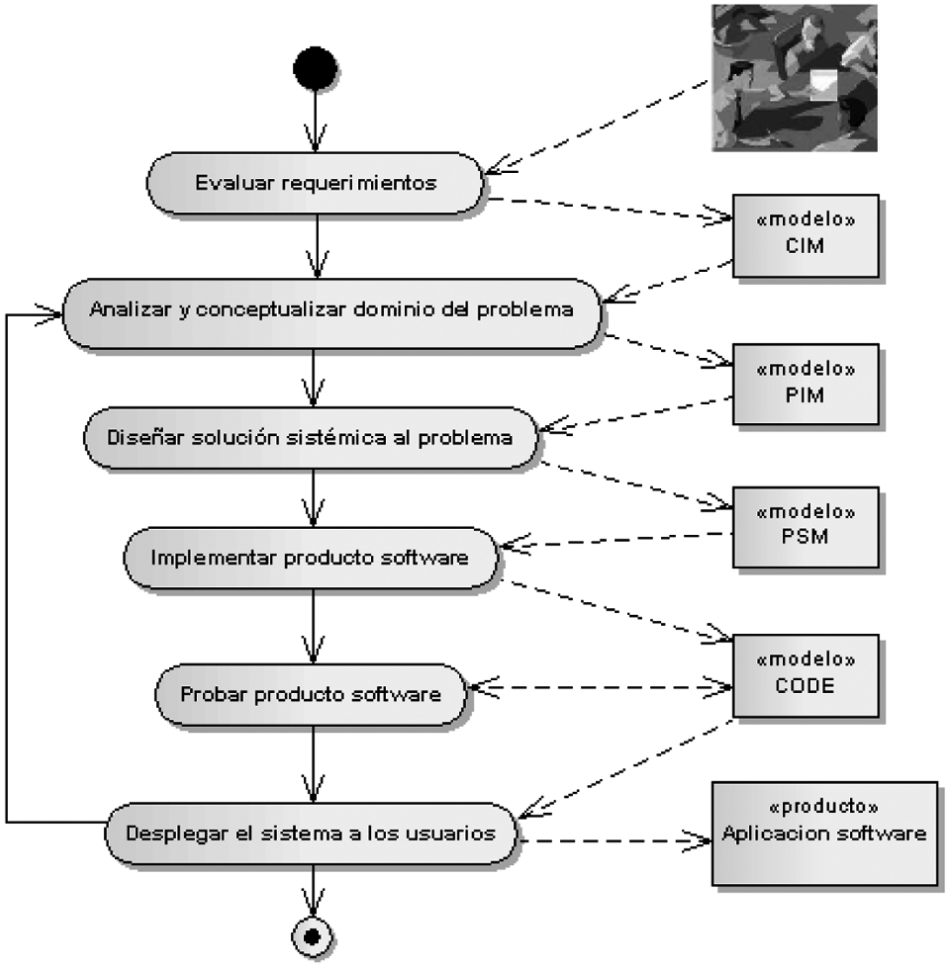

Figura 1: Proceso para la producción de productos software en la línea de trabajo MDD.

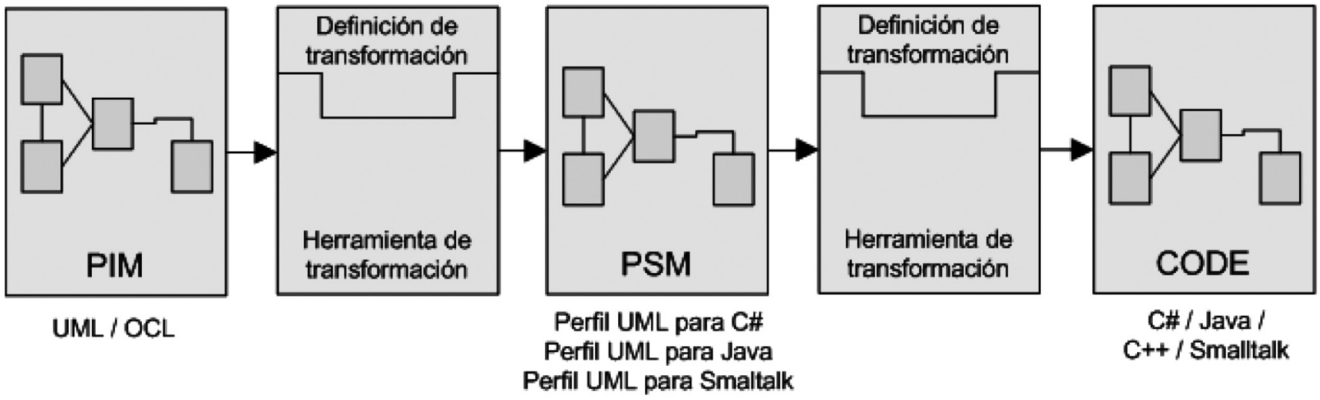

Figura 2: Evolución y transformación de modelos en el Proceso MDD

específicas ligadas a una tecnología concreta. A partir de un PIM se procede a la generación automática de uno o más PSMs los cuáles son la proyección del PIM en una plataforma específica. De esta manera podemos obtener múltiples PSMs para diferentes tecnologías de implementación, como ser: java, C++, C#, Python, etc.

CODE ó **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 en una plataforma de desarrollo.

Como se detalló, los modelos se suceden desde el más abstracto hasta el más concreto. En principio se procede en la elaboración del CIM (visión general del dominio e integración de información relevante para la definición de los requerimientos sistémicos) y a partir de éste se crea el PIM (solución lógica del problema). Obviamente no es posible realizar esto automáticamente ya que es impracticable 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 o los PSM (una solución lógica del problema a resolver definida a través de un PIM podría ser evaluado a través de distintos lenguajes de programación - C#, Java, Smalltalk, etc. – generando un PSM para cada caso) mediante transformaciones automáticas provista por potentes herramientas de automatización creadas para tal fin y a partir de un PSM específico, obtener también mediante transformaciones, su correspondiente modelo CODE (código fuente compilable para un lenguaje de programación específico) de la aplicación (producto software). Es importante destacar la posibilidad de generar desde un mismo PIM, PSMs orientados a distintas plataformas de implementación, lo que 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 automáticamente mediante herramientas CASE dedicadas, 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 CODE respectivamente. En general el modelo del cual se parte tiene un nivel de abstracción mayor que el resultante de la transformación.

Dada la importancia del modelado, en MDD es necesaria la utilización de lenguajes de modelado sustentados por una definicón formal que permita la construcción de modelos precisos, comprensibles, consistentes y completos. En la actualidad el lenguaje gráfico para modelado de mayor uso es UML (Unified Modeling Language) el cual posee una sintáxis gráfica amigable. La base formal necesaria para la aplicación de UML sobre MDD fue provista por el consorcio internacional OMG (Object Management Group) a partir del metalenguaje MOF ((Meta Object Facility, 2012). Para describir reglas de buena formación de los modelos, reglas de negocio concretas y proveer de mayor precisión a modelos escritos en lenguajes gráficos (modelos en UML), el OMG estandarizó un lenguaje formal (fuertemente textual y no gráfico como UML) basado en lógica de primer orden llamado OCL (Object Constraint Language) (Warmer and Kleppe, 2003).

En lo que se refiere a la construcción y control

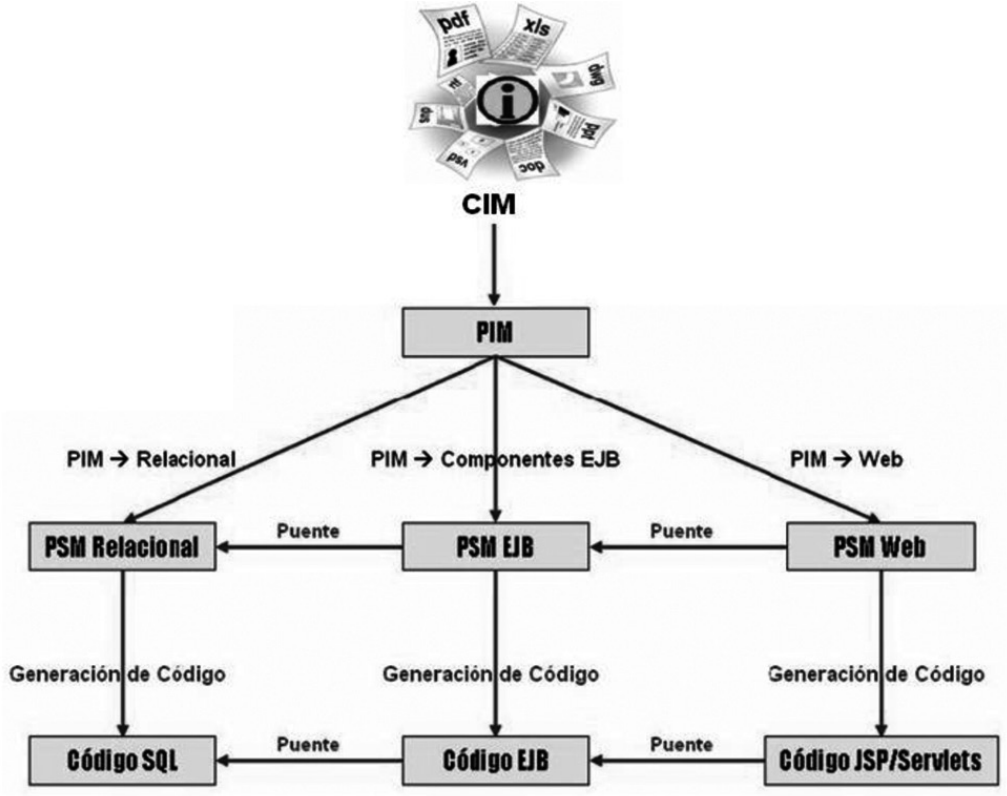

Figura 3: Ciclo de vida del Proceso MDD aplicado a varios PSMs.

de los diversos modelos, será fundamental el uso de aplicaciones, llamadas herramientas CASE (Computer Aided Software Engineering) (Agile Model Driven Development (AMDD), 2012), que proveen utilidades para la edición de modelos y su documentación, análisis de impactos frente a cambios, estimación de costos, 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 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).

Planificación y administración de un proyecto basado en 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 construídas 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 caracter estático sino que, 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 ésto 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 sólo 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 de características semejantes.

Reuso de artefactos: beneficios

Como en cualquier otro proceso productivo, el ahorro por medio de reusabilidad sobre ciertos elementos/procedimientos es una propiedad muy deseada y valorada, ya que nos permite reducir costos 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 reuso de los modelos y las transformaciones tiene mucho valor, ya que en ellos se encuentran incorporados el conocimiento y la experiencia de los especialistas que estuvieron a cargo de su construcción en toda la línea productiva, quedando ésta disponible para su uso aún después de que estas personas dejen de formar parte del proyecto.

Dadas las características inherentes a un proyecto dirigido por modelos, el reuso es una propiedad que no debe desaprocharse. 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.

ENFOQUES DE PROCESO LIVIANO PARA PRODUCCIÓN DE 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 el feedback con los desarrolladores es muy bajo o nulo.

En contraposición a esta visión surgen metodologías iterativas, dinámicas, adaptativas y centradas fuertemente en las relaciones con el usuario, llamadas en la industria de software actual *metodologías ágiles*. 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 de tiempo (iteraciones) en los cuales se llevan acabo 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 últimos años se dieron a conocer diversas metodologías ágiles como: XP (eXtreme Programming), TDD (Test Driven Development), Feature Driven Development (FDD), SCRUM, DSDM (Dynamic Systems Development Method), AUP (Agile Unified Process), AM (Agile Modeling), ASD (Adaptive Software Development), entre otras.

PROCESOS ÁGILES EN AMBIENTES DE PRODUCCIÓN MDD

Se han expuesto las obvias ventajas de la aplicación de una filosofia ágil para el desarrolo de software y por ende es importante adaptar estos conceptos al paradigma MDD. Una de las ideas principales adoptadas por la mayoría de las metodologias ágiles es la división del proyecto en iteraciones cortas. Esto nos permite disminuir la incertidumbre manteniendo unos 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 crítica de los usuarios.
- Se alcanzan resultados rápidamente con un esfuerzo mínimo, de manera que 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 manera que 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 las prácticas y experiencia ganada.

Los conceptos del desarrollo dirigido por modelos son muy recientes y todavía no han sido adoptados completamente por el campo informático, menos aún por la industria en su totalidad (aunque existen varios desarrollos de proyectos reales que aplican la visión productiva MDD). 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 y 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. Aún así las metodologías ágiles parecen lo más apropiado a la hora del desarrollo dirigido por modelos. De esta manera podemos aplicar los conceptos ágiles comentados previamente al modelado del PIM. Es decir, bajo una filosofía ágil de desarrollo de software en un contexto MDD, se desarrollan modelos de manera rápida desde las primeras iteraciones, se privilegia la participación activa del usuario, se fomenta la comunicación entre los individuos y demás beneficios que hemos comentado.

El informático Scott W. Amber ha impulsado y promovido conceptos propios de las metodologías ágiles para la construcción de software centrándose específicamente en el modelado. Amber propuso una innovadora versión de MDD, denominada Agile Model Driven Development (AMDD), (2012), 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.

CONCLUSIONES

Dada la creciente popularidad de las metodologías ágiles para el desarrollo de software en proyectos de pequeña y mediana envergadura, la aplicación de procesos ágiles para la producción de software en ambientes dirigidos por modelos se presenta como un recurso favorable para acercar este paradigma tanto a aquellos equipos de desarrollo familiarizados con los conceptos ágiles que desean aprovechar las ventajas provistas por MDD, como a aquellos equipos que ya utilicen MDD y deseen obtener los beneficiosos de los conceptos ágiles.

Como líneas de trabajo futuro se encuentra en consideración: la evaluación de herramientas CASE que den soporte a la edición y transformación de modelos mínimos para un ambiente de producción ágil de productos software, evaluar la aplicación de patrones de modelado para agilizar la construcción de modelos CIM y PIM (ya que son los elementos de entrada que mayor tiempo y esfuerzo requiere en el proceso de producción en MDD) y estudio de lenguajes gráficos para el modelado CIM que den soporte a la etapa inicial (que requiere mayor nivel de creatividad) del proceso MDD estandarizado a través de técnicas/herramientas/tecnología de MDA.

REFERENCIAS

Agile Model Driven Development (AMDD) (2012).- www.agilemodeling.com

Grady Booch, I. J. y Rumbaugh, J. (2006). "El lenguaje unificado de modelado (UML)". Segunda Edición. Pearson Education, S.A. - ISBN-13: 978-847-82-9076-5

Meta Object Facility (MOF) (2012).- www.omg.org/mof

Model Development Architecture (MDA) (2012).- www.omg.org/mda

Object Management Group (OMG) (2012).- www.omg.org

Pons, C., Giandini, R. y Perez G., (2010). "Desarrollo de Software Dirigido por Modelos." Conceptos teóricos y su aplicación práctica 1er. Edición. EDULP & McGraw-Hill Educación. Argentina ISBN-13: 978-950-34-0630-4 - http://www.agilealliance.org/

Sommerville, I., (2005). "Ingeniería del Software. Séptima edición". Pearson Education, S.A. ISBN: 84-7829-074-5

Warmer, J. and Kleppe, A. (2003) "The Object Constraint Language. Getting Your Models Ready for MDA". Pearson Education, S.A.