Revisión sistemática de la integración de modelos de desarrollo de software dirigido por modelos y metodologías ágiles

Systematic review about the integration of model-driven software development and agile methodologies

Recibido: 30- 04 - 2016 Aceptado: 26-05-2016

Juan José Morales Arias[1]
César Jesús Pardo Calvache[2]

[1] Colombiano. Ingeniero de Sistemas. Especialista en desarrollo de software por la Universidad EAFIT, con experiencia en el desarrollo de software y metodologías ágiles soportadas sobre SCRUM, experiencia en tecnologías como J2EE, Oracle PL\SQL, entornos LAMP, WEB con AngularJS, GWT y Struts. Correo electrónico:jmoral24@eafit.edu.co

[2] Colombiano. PhD y MSc. en Ingeniería Informática por la Universidad de Castilla-La mancha, España. Profesor Asistente del Departamento de Sistemas, Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca. Calle 5 No. 4-70.C.P. 19002. Popayán, Colombia. Correo electrónico: cpardo@unicauca.edu.co

Resumen

Actualmente, en algunas instancias, la industria de desarrollo de software se lleva a cabo por medio de actividades manuales y/o metodologías robustas que pueden llegar a ser en muchos casos pesadas e ineficientes. Esta situación trae consigo algunos problemas relacionados con la dificultad para producir software de manera oportuna, ágil, a bajo costo y con un alto nivel de calidad. Una manera de mejorar esta situación está en añadir al proceso de desarrollo de software el formalismo y la abstracción necesaria que permita automatizar y optimizar las tareas más críticas definidas, a partir de las metodologías utilizadas en las empresas de software, y desde una perspectiva ágil. Esto añadiría valor agregado a los negocios y mejoraría el proceso de software considerablemente. En este sentido, con el objetivo de conocer las bondades de los enfoques ágiles y los entornos de programación dirigidos por modelos, se llevó a cabo una revisión sistemática de la literatura en relación con los proyectos donde se integran estos enfoques a nivel mundial, así como la identificación de los beneficios declarados por diferentes estudios.

Palabras clave: desarrollo de software dirigido por modelos, arquitectura dirigida por modelos, ágil, proceso software, desarrollo.

Abstract

Currently, in some instances of the software development industry are carried out by means of manual activities and/or robust methodologies which can be often heavy and inefficient. This

situation brings several issues related to the difficulty to produce software in a timely manner, agile, at low cost and with a high quality level. A way to improve this situation is to incorporate in the software development process the formalism and abstraction needed to automate and optimize the most critical tasks defined from methodologies used in software companies and starting from an agile approach. This would add value to the business and would improve significantly the process of software. In this sense, in order to publicize the benefits of agile approaches and programming environments driven models, a systematic review of the literature has been conducted so as to the projects where these approaches have been integrated globally. Besides, it has been possible to identify some benefits, which have been reported by different studies.

Keywords: Model driven software development **(**MDSD), model driven architecture (MDA), agile, software process, development.

Introducción

Los primeros lenguajes de programación como Assembler (Batson et al., 2015), permitían escribir programas en los cuales se pueden mover datos a direcciones de memoria específicas y ejecutar operaciones propias del procesador; sin embargo, en este punto el nivel de abstracción estuvo más cercano a la máquina y lo distanció del lenguaje en el que se desarrollan normalmente los problemas humanos. Después de los lenguajes de bajo nivel, surgieron lenguajes procedimentales más estructurados, por ejemplo: FORTRAN (Chivers & Sleightholme, 2016) y el lenguaje C (Kernighan & Ritchie, 1988) donde era posible definir conjuntos de instrucciones con nombres cercanos a la realidad, que se trataban de representar y llamarlos en un orden lógico dentro de la estructura de un programa. A finales de los 80, aparecieron los lenguajes orientados a objetos que permitieron distanciarse un poco de los conceptos relacionados estrechamente con el computador (hardware), para programar pensando más en las características, acciones y relaciones de los objetos que estaban siendo analizados (Terren, Moreno, & Jimenez, 2007). En este punto, el nivel de abstracción y el lenguaje con el cual representamos la realidad en un programa de computadora había evolucionado considerablemente en comparación a su comienzo.

A principios del siglo XXI se desarrolló el Lenguaje Unificado de Modelado (UML), lenguaje de modelamiento que aumentó el nivel de abstracción utilizado para representar la realidad de los sistemas informáticos que eran desarrollados (Object Management Group, 2015). Este lenguaje fue integrado a los procesos de desarrollo de software que habían logrado niveles de aceptación en la industria del software, por ejemplo, Rational Unified Process (RUP) o su variante ágil Agile Unified Process (AUP), utilizan este lenguaje de modelado en sus etapas para representar aspectos diferentes del sistema que son relevantes en las disciplinas definidas en cada una de las fases. En esa época, incluso se desarrollaron herramientas que permitían generar código de software a partir de los modelos construidos (Objects by Design, 2016). Sin embargo, el código generado era demasiado básico, y por lo tanto, los modelos solo eran considerados útiles en las etapas tempranas del desarrollo de software, es decir, en las etapas de análisis y diseño, y el código generado era desechado o simplemente no se utilizada dado que su adaptación y modificación, en algunos casos, era más costoso que desarrollar una aplicación desde cero (Andrade, Ferreira, & Sinderen, 2004).

En este sentido, algunos investigadores empezaron a interesarse por la implementación de metodologías enfocadas a la utilización de modelos y a su trasformación, de acuerdo a ciertos criterios definidos a través de un lenguaje específico de dominio (Domain Specific Language - DSL) en software ejecutable (Haase et al., 2007; The eclipse Foundation, 2016 ; Völter et al., 2006). Con este objetivo se pretendió mejorar la productividad en la industria del desarrollo de software, y aumentar el nivel de abstracción a niveles donde el software y los modelos que lo representan estén acoplados y se puedan cambiar a la misma velocidad que lo hacen los negocios.

Sin embargo, aún no hay un uso lo suficientemente extendido del desarrollo de software dirigido por modelos (Model Driven Software Development - MDSD) y no se logró identificar directrices claras sobre su implementación a nivel general, ya que los esfuerzos que se han realizado en este sentido son aislados y denotan un nivel de inmadurez que ha impedido su industrialización (Bernardo-Quintero & Duitama-Muñoz, 2011).

Por otro lado, la industria del software está tendiendo a integrar e institucionalizar procesos ágiles y menos complejos que le permitan adaptase a los cambios del negocio en forma oportuna, y que además permitan agregar valor a los usuarios en el menor tiempo posible (Fallas, 2012; Zhang & Patel, 2011). Teniendo en cuenta que

lo anterior es afín a los objetivos de MDSD, en el sentido en que se enfatiza en la capacidad de cambio de los sistemas con respecto a la realidad que pretenden representar y a la entrega temprana de productos de valor, se han planteado algunas propuestas desde diferentes ópticas, cuyo fin es tomar ventajas de estas dos metodologías y formular un proceso de desarrollo donde se aplique lo mejor de estos dos enfoques.

La revisión sistemática ofrece una perspectiva de las iniciativas desarrolladas en torno al desarrollo de software, dirigido por modelos dentro del contexto de las metodologías ágiles, analizando qué aplicaciones prácticas se han llevado a cabo y cuáles han sido sus resultados.

El documento se organizó de la siguiente manera: la primera parte muestra una descripción detalla del proceso de la revisión sistemática incluyendo la formulación de la pregunta, selección de las fuentes y estudios, y la extracción de la información. Seguido; muestra los resultados obtenidos en el estudio con base en el análisis de los resultados. Finalmente, se presentan las conclusiones y trabajo futuro.

Diseño de la revisión sistemática

Para la realización de la revisión sistemática del desarrollo de software dirigido por modelos y las metodologías ágiles, se siguió la plantilla de protocolo utilizada en (Biolchini, et al., 2005) y el protocolo para la revisión presentado en (Brereton et al., 2008), el cual se enfocó principalmente en: (i) la formulación de la pregunta, (ii) selección de las fuentes, (iii) selección de los estudios, (iv) extracción de la información, y (v) resumen de los resultados. La Figura 1 muestra el diagrama de flujo de las actividades realizadas en el proceso de revisión.

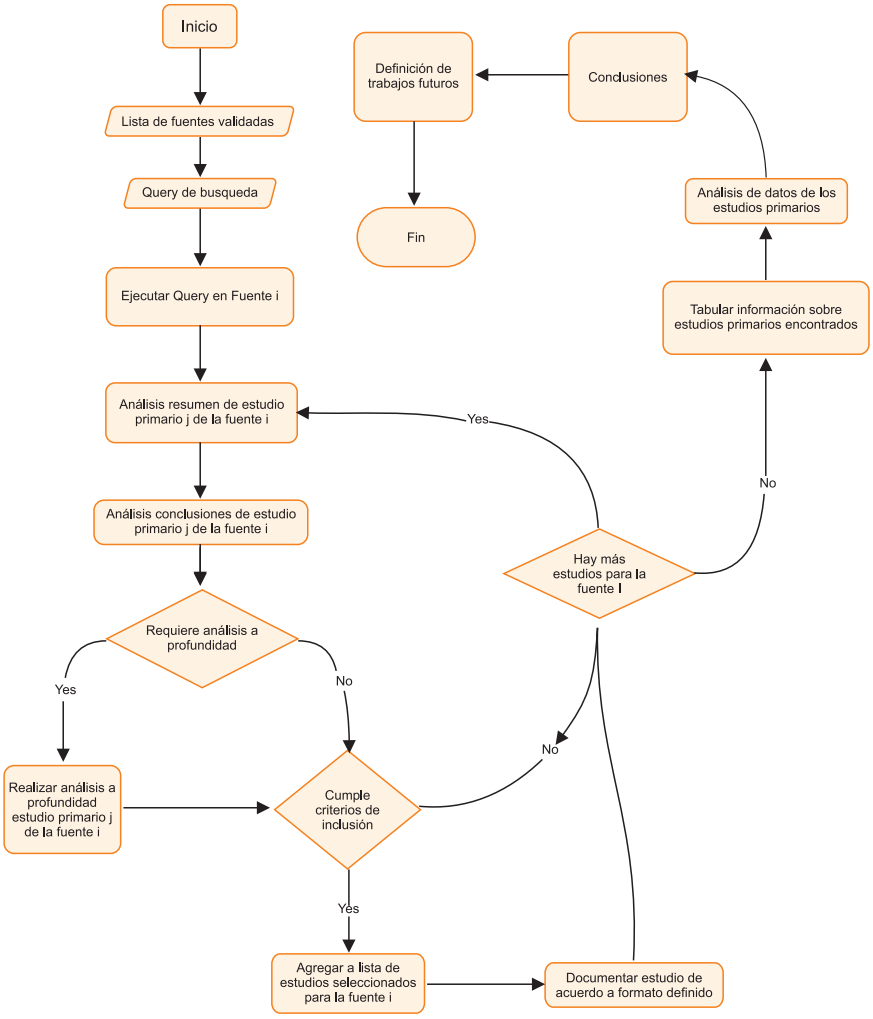

Figura 1. Protocolo de la revisión sistemática
Fuente: los autores.

Formulación de la pregunta de investigación

La pregunta de investigación que se definió para la realización de esta revisión sistemática fue: ¿Qué trabajos e iniciativas relacionadas con el Proceso de desarrollo de software dirigido por modelos y su aplicación en metodologías ágiles se han llevado a cabo? La Tabla I muestra el listado de términos utilizados para diseñar la pregunta de investigación.

En el contexto de la revisión sistemática planeada se observaron y analizaron las propuestas y trabajos existentes relacionados con el Proceso de desarrollo de software dirigido por modelos y su aplicación en entornos de desarrollo ágiles. Como mecanismo de control, se revisaron los trabajos relacionados, entre ellos algunos libros que profundizan en la aplicación práctica y algunas actividades que se llevan a cabo en forma genérica. De estos trabajos se han obtenido las palabras clave, y a pesar de que no se hayan encontrado la mayoría de estas publicaciones en las fuentes de búsqueda seleccionadas al momento de la revisión sistemática, se ha considerado importante tenerlos como referencia dentro del conjunto de los estudios primarios como literatura gris por su relevancia y relación a la revisión sistemática.

La población objetivo que se estudió en la revisión sistemática se representó en las publicaciones de las fuentes de datos seleccionadas y se relacionaron con el objetivo de la revisión sistemática. En la Tabla 1 se presentan los términos utilizados.

Tabla 1. Términos utilizados

Términos	
Palabras clave	**Sinónimo**
Process	Procedure, developed, technique
Driven	Guided,based,oriented
Model	pattern, type, archetype
Software	System
Agile	Rapid
Activities	Action, steps
Architecture	Design, definition, structure

Fuente: los autores.

Selección de las fuentes

Con el listado de palabras clave y la utilización de los conectores lógicos "AND", "OR" y "NOT", se realizaron las siguientes estrategias de búsqueda:

(process OR procedure OR techniques OR development) AND (model OR patterns OR template) AND (architecture OR design OR structure) AND (driven OR guided OR based OR oriented).

AND (software OR system) AND development AND (activities OR actions OR steps) AND (agile OR scrum OR xp).

En la revisión sistemática se emplearon las siguientes fuentes: Wiley Online Library en el tema de Computer Science, ACM Digital Library, ProQuest y la IEEE Computer Society, los sitios oficiales del Manifiesto Ágil, SCRUM, eXtreme Programming (XP), y otros artículos relacionados con el tema se revisaron como literatura gris. En el momento de realizar la búsqueda estratégica, se adaptó a las características de cada uno de los motores de búsqueda de las fuentes escogidas.

Selección de los estudios

Definidas las fuentes de información primaria, se describió el proceso de selección y análisis de los estudios primarios resultados de la búsqueda, para así determinar su pertinencia y aporte relevante a la revisión sistemática.

La selección de estudios se basó en un proceso iterativo incremental, que permitió llevar a cabo la búsqueda y extracción de los resultados en cada una de las fuentes. El proceso incremental se desarrolló debido a que el proceso de búsqueda y análisis se realizó de forma sucesiva o iteradamente en cada una de las fuentes de búsqueda, de esta forma la información se incrementó a medida que se agregaron datos resultantes del análisis de los estudios seleccionados.

Extracción de la información

Los criterios de inclusión de estudios primarios se enfocaron principalmente en el análisis de los siguientes atributos: (i) título, (ii) resumen, y (iii) las conclusiones de cada estudio. Se buscó identificar en qué medida se proponían o evaluaban los modelos para la implementación de procesos de desarrollo dirigidos por modelos orientados a metodologías de desarrollo ágil. Para determinar la pertinencia de los artículos, se hizo necesario hacer un análisis de su contenido en profundidad, teniendo en cuenta variables como el nivel de descripción del modelo propuesto, su relación con el dominio de metodologías ágiles, y su pragmatismo.

Cómo criterio de exclusión se tuvo en cuenta, luego de una revisión a profundidad, los artículos que hacían un

estudio o diagnóstico general de la metodología, pero que no definían o proponían un modelo para su aplicación en forma concreta.

La documentación de los estudios primarios seleccionados se estructuró en una tabla con el fin de definir las variables concretas de interés y en la cual se tuvo en cuenta la siguiente información: código, titulo, subtitulo, fuente, país, año, autores, palabras clave, referencias (estudios secundarios), ideas principales, tipo de proceso propuesto y aplicaciones. La Tabla 2 muestra un resumen del formato usado para representar la información relevante.

Tabla 2. Ejemplo de la plantilla utilizada para organizar la información de un estudio primario

Título	Towards Patterns for MDE-Related Processes to Detect and Handle Changeability Risks
Subtítulo	No hay información.
Fuente	ACM
País	ALEMANIA
Año	2012
Autores	Regina Hebig, Gregor Gabrysiak, and Holger Giese
Palabras clave	Proceso de desarrollo de software, análisis, control de cambios, patrones
Ideas Principales	
En el presente artículo se hace un estudio detallado sobre los patrones que pueden aplicarse en casos donde sistemas desarrollados bajo la metodología MDSD tienen repetidos cambios a través del tiempo. Se plantean las diferentes formas en las que se puede afectar la integridad del sistema y se propone una notación para representar los diferentes casos en los que se puede presentar estos riesgos, además de 4 posibles aplicaciones que permiten mitigarlos. El proceso de desarrollo de software tiene un impacto directo en la productividad y en la calidad de los productos de software. De igual forma, las mejoras en la productividad y la calidad logradas a través de Model Driven-Engineering (MDE) tienen que ser el reflejo de un buen modelo que soporte la dinámica del desarrollo de software en contextos de negocio que cambian fácilmente en el tiempo. Se descubrieron tres escenarios en los que un proceso de MDE debe controlar la capacidad de cambio del sistema: • Pérdida o preservación del contenido del artefacto. • Pérdida de relaciones explicitas entre los diferentes artefactos. • Actividades automáticas que pueden resultar enganchadas de forma que dos cambios no puedan dar se uno independiente del otro.	

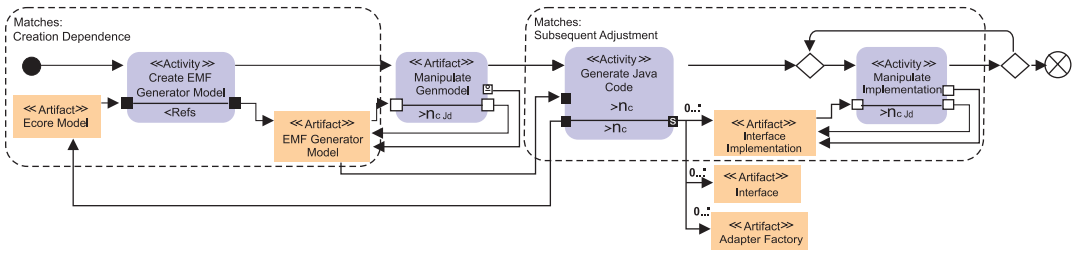

Figura 3. Exemplary process of the EMF case study with initially perceived SwMaMo activities

Se plantea además el siguiente conjunto de patrones para manejar cada situación:

1. Process Proto Pattern Subsequent Adjustment: Se usa en el contexto en que se presenta pérdida inesperada de contenido. La creación actividad inicial de forma automática crea artefacto ajustado y lo llena de contenido en la base de artefacto base (es decir, tanto los artefactos solapan en contenido después de la ejecución). De este modo, ya no existente versión que se considera artefacto ajustado. El ajuste de la actividad agrega aún más el contenido de artefacto ajustado manualmente.

2. Process Proto Pattern Creation Dependence: Este es un antipatrón que hace referencia a la dependencia de referencias entre artefactos. Donde un artefacto de entrada es a la vez un artefacto de salida, creando así una dependencia cíclica.

3. Process Proto Pattern Split Manufacture: Este patrón hace referencia al caso de pérdida inesperada de contenido en un artefacto. Aquí se propone conservar el detalle de artefacto antes y después de ser regenerado, y ajustar manualmente los detalles de este.

4. Process Proto Pattern Anchor: Abarca en el caso de pérdida de relaciones explicitas entre artefactos. Aquí cada referencia tiene un ancla en el artefacto que guarda la referencia correspondiente a este en cada regeneración o cambio manual.

Tipo de proceso propuesto
Se propone un conjunto de actividades, patrones y antipatrones para abarcar riesgos que se dan, debido a la capacidad de cambio del sistema en un contexto MDE.
Aplicaciones
Se hace un caso de estudio con un equipo de desarrollo de SAP utilizando EMF y hacen una comparación con las actividades y patrones propuestos.

Fuente: los autores.

Resultados y discusión

Con la información extraída se realizó un análisis de los datos obtenidos, teniendo en cuenta los resultados generales y específicos después de haber sido aplicados los criterios de inclusión y exclusión.

Tendencias de las publicaciones

Luego de aplicar el protocolo para obtener los estudios primarios, se encontraron 42 estudios relevantes relacionados con desarrollo dirigido por modelos, de los cuales 22 tenían relación específicamente con procesos MDSD, orientados a las metodologías ágiles. Luego de un análisis detallado, se seleccionaron 10 estudios primarios que fueron evaluados bajo criterios de inclusión y exclusión, donde se evaluó en qué medida proponían un proceso específicamente para la aplicación de MDSD en entornos de desarrollo ágil. La Figura 2 muestra la tendencia de las publicaciones desde el año 2000 al 2014. Como se puede observar, hay un creciente interés en el área de MDSD en la última década, concentrándose un mayor interés entre los años 2006 y 2012, donde se concentran el 50% de los estudios encontrados relacionados en relació con este tema. Este indicador nos permite establecer que hay un interés marcado de la industria y la academia por este enfoque de desarrollo de software, y que se están adelantando trabajos a nivel teórico y práctico. Por su parte, no hemos encontrado estudios relevantes para el 2015, quizá, porque aún no se termina el primer semestre del año.

Figura 2. Tendencia de las publicaciones por año
Fuente: los autores.

Distribución por país

La Figura 3 muestra la distribución en orden descendente de los países que reportan más estudios realizados en relación a MDSD, estos son: Estados Unidos, Brasil, España, Alemania, Brasil y China. Con relación a Brasil, China y Kenia muestran interés en la formalización de la industria del software, además, han realizado estudios significativos que abordan el tema específico de MDSD, por ejemplo, en Kenia se desarrolló un modelo Adaptable de MDA (Arquitectura dirigida por modelos) para métodos formales y su integración con metodologías ágiles (Rigworo, 2013), en Brasil se realizó un proceso de desarrollo dirigido por modelo orientado a las pruebas (Almeida & Oliveira, 2014), y en China se diseñó un enfoque conceptual para modelar el proceso de desarrollo de software dirigido por modelos (Duan & Fu, 2006).

Lo anterior permitió observar los diferentes puntos de vista relacionados a las propuestas y soluciones desarrolladas, esto trae consigo un amplio abanico de posibilidades que se pueden implementar en diferentes tipos de empresas relacionadas con desarrollo de software, donde se aplicó una arquitectura de software genérica para diferentes dominios de negocio, y donde se tenga un enfoque orientado a la entrega continua de productos. Asimismo, permite observar que incluso industrias emergentes están viendo el MDSD como una opción aplicable en sus contextos. Por otro lado, la diversidad en la procedencia de los estudios primarios muestra que este enfoque está adquiriendo interés cada vez más en la industria del software, y que no es una tendencia aislada en ciertos sectores de investigación y desarrollo privilegiados, por ejemplo:

Existen herramientas para la generación de aplicaciones web dirigida por modelos que se vinculan con procesos de negocio y que se han implementado con éxito en entornos empresariales (Kraus, Knapp, & Koch, 2007; Kroiss, Koch, & Knapp, 2009).

Se han implementado generadores de ETL (Procesos de extracción, transformación y carga para análisis de datos) dentro de procesos de inteligencia de negocios que permiten, a partir de un modelo de base de datos, generar extracciones de datos de interés para crear y poblar un modelo intermedio con base en el cual generar consultas que aporten información valiosa para la toma de decisiones (Oracle, 2015; El Akkaoui, Zimànyi, Mazón, & Trujillo, 2011).

Existen herramientas que generan interfaces gráficas Web a partir de modelos, y permiten a su vez administrar modelos de datos que son generados a través de modelos de negocio. En este caso tenemos, por ejemplo, a Open Xava (Paniza, 2011), Moskitt (Hernández, León, & Ferrara, 2015), Spring Roo (Sarin, 2011), entre otros.

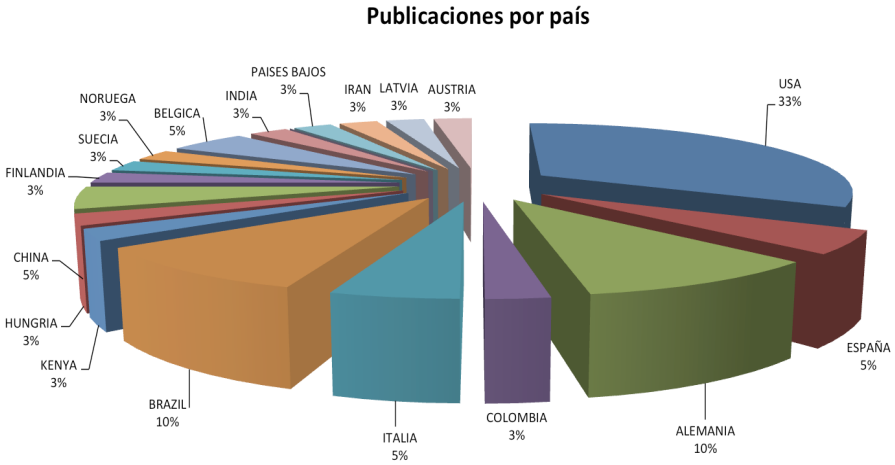

Figura 3. Publicaciones por país.
Fuente: los autores.

Es importante resaltar que el fortalecimiento de la industria de software depende directamente de la ayuda del Estado, universidades y empresas. En Colombia, el Ministerio de Tecnologías de la Información y la Comunicación (TIC) (MinTIC, 2016), y otros organismos a nivel nacional y departamental como el SENA (SENA, 2016), INNPULSA (INNPULSA, 2016), RED CLUSTER Colombia (RED CLUSTER COLOMBIA, 2016), (Ruta N, 2016), Parquesoft (Parquesoft, 2016), Bancoldex (Bancoldex, 2016), Fedesoft (Fedesoft, 2016) (Market, 2012), Intersoftware (Intersoftware, 2016), clusters, entre los que se destacan PacifiTIC (PacifiTIC, 2016), CaribeTIC (CaribeTIC, 2016), SinerTIC (SinerTIC, 2016), ClusterTIC Antioquia (ClusterTIC, 2016), Apps.co (Apps.co, 2016), Vive Digital, cámaras de comercio, entre otros, han impulsado iniciativas que han permitido fortalecer el sector TIC en Colombia. Sin embargo, los temas en materia del desarrollo dirigido por modelos en Colombia son aún bastante lejanos, quizá, porque se desconocen los beneficios asociados a este.

Beneficios identificados de la implementación de MDSD en empresas

A partir del análisis de los estudios primarios, ha sido posible identificar algunos beneficios de la implementación de prácticas de desarrollo de software dirigida por modelos. La Tabla 3 presenta un conjunto de los beneficios identificados.

Tabla 3. Beneficios de la aplicación de MDSD dentro de procesos de desarrollo de software

	Beneficio	Descripción
1	Calidad	El software generado bajo el mismo lenguaje específico de dominio (DSL) tendrá la misma arquitectura de referencia y no estará expuesto a errores humanos inherentes a la complejidad del código que tendría que ser escrito a mano, en caso de que el proceso se hiciera en forma manual.
2	Migración y generación de código en múltiples lenguajes - Portabilidad	Se puede migrar un producto fácilmente a la versión más reciente de la tecnología en la que fue implementado o a otra tecnología diferente. Esto es posible, ya que en este caso solo es necesario cambiar los mecanismos de trasformación modelo a texto (Model to Text - M2T) sin afectar el modelo. De esta manera, un producto que actualmente está implementado en lenguaje PHP podría ser migrado fácilmente a otros lenguajes, por ejemplo, Ruby o Python cambiando solamente las reglas de trasformación para cada lenguaje específico.
3	Integración	A partir del modelo de implementación se puede generar automáticamente un modelo de pruebas que a su vez genere automáticamente un conjunto de pruebas unitarias ejecutables. De igual manera, se puede modelar también procesos de integración continua, en los que se definan los procedimientos para desplegar la aplicación en un ambiente específico.
4	Productividad	Aunque los esfuerzos iniciales se concentrarán en tener un lenguaje específico, lo suficientemente robusto para soportar la necesidades del negocio, este enfoque permite aumentar la productividad del equipo de desarrollo paulatinamente, mejorando el tiempo de entrega de los diferentes entregables definidos. Esto, debido a que lo que antes requería mayor cantidad de tiempo en ser codificado será autogenerado a través de las generaciones modelo a texto (M2T).
5		A través de la automatización se puede generar código ejecutable a partir de modelos formales, usando uno o varias fases de trasformación.
6	Separación de ámbitos	Los aspectos del sistema que no pueden ser tratados fácilmente en un solo módulo, pueden variar y afectar los módulos que dependen de este. A través de este enfoque se garantiza la mantenibilidad y disminución de la redundancia del código generado incluyendo sus dependencias.
7	Reutilización	Una vez definidos los DSL (Lenguajes Específicos de Dominio) correspondientes a la arquitectura y al negocio, se pueden reutilizar en diferentes aplicaciones que usen la misma arquitectura o estén relacionadas con el mismo ámbito de negocio.
8	Control de la complejidad a través de la abstracción	Los DSL permiten llevar los problemas a niveles de complejidad más básicos y cercanos al mundo real, encapsulando su complejidad técnica a los procesos de transformación que tendrían que hacerse solo una vez.
9		MDSD sigue los mismos lineamientos de OMG (Object Mangement Group) [46], donde se establece la independencia y estandarización de los modelos. Esto permite que se pueda inter operar con otras tecnologías que sigan los mismos estándares.

Fuente: los autores.

Soluciones propuestas

La Tabla 4 muestra una clasificación de los procesos propuestos, de acuerdo a su enfoque y relación con metodologías de desarrollo de software convencionales. Asimismo, se pudo observar que un 50% de los estudios primarios proponen o aplican procesos en los cuales se lleva a cabo la integración de enfoques y prácticas de ingeniería dirigida por modelos o MDE y metodologías ágiles (Cockburn, 2006 ; Canós, Letelier, & Penadés, 2003). También se observa que MDSD ha sido integrado al RUP (Kroll, Kruchten, & Booch, 2003), donde el objetivo ha sido integrar las actividades de MDE a las etapas de diseño y elaboración del RUP (Kroll et al., 2003). Otras soluciones están dirigidas a integrar metodologías como Model Driven Testing (MDT) dentro de un proceso en espiral e incremental (Almeida & Oliveira, 2014), o en buscar formas de controlar de manera eficiente y a través de patrones, los cambios en los modelos que pueden afectar el código generado en diferentes escenarios (Hebig, Gabrysiak, & Giese, 2012) y en metodologías para la trasformación modelo a modelo (M2M) (The Eclipse Foundation, 2012), o en generación de diferentes capas y niveles de abstracción de código a partir del mismo modelo (El Akkaoui et al., 2011).

En la Tabla 5 se muestra un análisis a nivel general de las propuestas encontradas en los estudios analizados, su relación con otros enfoques y metodologías, y una pequeña descripción de sus ventajas. A partir del análisis se identificaron algunas tendencias y enfoques en los procesos propuestos. En primer lugar, existe una tendencia marcada a comparar las prácticas de MDE, incluyendo sus actividades y principios a procesos formales como RUP, Scrum y XP. En esta tendencia se destacan procesos como MDD-SLAP (Zhang & Patel, 2011), que al igual que en la metodología de gestión ágil Scrum, también está conformado por *sprints*, con un conjunto de iteraciones que describen actividades propias tanto de MDE como de otras áreas de la ingeniería del software, como el aseguramiento de la calidad e integración continúa. Por otro lado, se encuentran procesos como SAGE (Matinnejad, 2011), cuyo principal objetivo es aplicar los principios del enfoque ágil, soportando la entrega de código ejecutable a partir de modelos conflictivos. Este proceso se vale de un conjunto de modelos desde diferentes enfoques que definen diferentes aspectos del sistema, como el comportamiento, el alcance y las interacciones, y a partir de trasformaciones entre estos modelos, generar un código ejecutable.

Como se pudo observar en la Tabla 3, hay una disminución de los métodos de desarrollo de software tradicionales, los cuales requieren una gran cantidad de documentación (incluyendo el código) creado a mano. Aunque, como se observa en la Tabla 4, al organizar los estudios por tipo de proceso propuesto, es posible encontrar que algunas propuestas se basan en procesos como RUP y otros en métodos formales, estos se enfocan en mapear las fases que corresponden a la implementación con las actividades propias de MDSD.

Tabla 4. Porcentaje por tipo de proceso propuesto

Cat.	Enfoque	Publicaciones	%
1	MDA - MDT en espiral	1	10%
2	Patrones MDE Manejo de cambios	1	10%
3	Trasformaciones multicapa	1	10%
4	Unificación con metodologías ágiles	5	50%
5	Unificación con RUP	1	10%
6	Aplicación de DSL a métodos formales	1	10%

Fuente: los autores.

Tabla 5. Principales procesos propuestos

Nombre del proceso	Orientación	Objetivo Principal	Ventajas	Año
SAGE (Kirby, 2006)	Basado en MDD.	Aplica orientación ágil para software altamente asegurado.	Soporta generación de ejecutables de modelos parcialmente conflictivos.	2006
Hybrid MDD (Guta, Schreiner, & Draheim, 2009)	Basado en Assembly.	Aplica orientación ágil para proyectos medianos y pequeños.	Aplicación parcial de actividades de MDD en colaboración con prácticas tradicionales de programación.	2009
MDD –SLAP (Zhang & Patel, 2011)	Basado en Agile.	Se beneficia de las ventajas de metodologías ágiles y MDD para desarrollar sistemas de comunicación en tiempo real.	Establece una correspondencia entre MDD y las prácticas ágiles.	2011
High Level Lifecycle (Ambler, 2004)(Ambler, 2002)	Basado en Agile.	Escala el desarrollo ágil.	Aplicación a gran nivel de MDD ágil.	2004

Fuente: los autores.

Conclusiones y trabajo futuro

La revisión sistemática realizada en el presente trabajo, reúne los resultados de los estudios primarios relacionados con procesos ágiles para el desarrollo de software dirigido por modelos. La revisión ha seguido un formalismo que ha permitido obtener una perspectiva completa del tema objeto de estudio. Asimismo, de validar los resultados obtenidos a partir del protocolo definido para llevar a cabo la revisión sistemática.

Los estudios seleccionados a través de los procesos definidos de inclusión y exclusión, permitieron obtener estudios relacionados con el tema de estudio. A través de dichos resultados, se logró identificar un significativo interés tanto por el enfoque de desarrollo de software orientado por modelos como por la adaptación de esta al enfoque y/o procesos de desarrollo con enfoque ágil. Con la identificación del crecimiento interés por integrar metodologías ágiles junto a enfoques dirigidos por modelos, es posible pensar que el impacto de estos dos enfoques en las micro, pequeñas y medianas empresas (mipymes) no está lejos del beneficio obtenido por grandes empresas que implementan el MDSD.

Con relación a las propuestas analizadas, es necesario destacar que cada uno de estos aborda un aspecto específico tanto del proceso de desarrollo como de su acercamiento al manifiesto ágil y aplicabilidad en la industria del software a gran escala. Sin embargo, existen algunos aspectos que no quedan claros, por ejemplo:

- ¿Se pueden integrar etapas del desarrollo de software como los requisitos, las pruebas y el despliegue dentro del marco de MDA?

- ¿Se puede mapear de la misma forma los procesos propuestos con modelos de madurez y capacidad para el desarrollo de software como CMMI?

- ¿Qué roles intervienen dentro de un proceso en el cual se pueda integrar una metodología MDE con metodologías ágiles? ¿Debe integrarse nuevos roles? ¿Cuáles y en qué etapas?

- ¿Existen herramientas que soporten la integración de estas metodologías y en todas las etapas del proceso?

- ¿Es posible aplicar este tipo de metodologías en pequeñas y medianas empresas en forma práctica y a bajo costo?

Este conjunto de incógnitas que surgen, permiten concluir que si bien hay un gran interés y desarrollos significativos para formular y aplicar procesos ágiles de desarrollo de software orientado por modelos, estos aún

se encuentran en una etapa inmadura, y que es necesario proponer soluciones con el fin de abordar un conjunto más amplio de aspectos del desarrollo de software a nivel industrial, que permitan aumentar el nivel de productividad y calidad en el desarrollo de software haciendo uso conjunto y síncrono de las metodologías MDD en el contexto de los procesos ágiles de desarrollo . Como se mencionó al comienzo del análisis de los resultados, no se han encontrado estudios relevantes para el 2015, quizá, porque aún no se termina el primer semestre del año 2016 y podría suponerse que trabajos relacionados aún no han sido indexados por las fuentes consultadas.

La industrialización de los procesos de desarrollo de software es necesaria, y se espera que no esté lejos de convertirse en un escenario posible al igual que en otro tipo de industrias. Es importante reconocer tanto los beneficios del MDSD, así como el esfuerzo que se debe realizar en la definición, adaptación e implementación de soluciones y estrategias que permitan mejorar los procesos productivos en la industria de software, más allá que hacerlo solamente a nivel de gestión como se realiza actualmente a través de la certificación de modelos mundialmente reconocidos como CMMI. El MDSD puede usarse como una solución para dar soporte al cumplimiento de buenas prácticas definidas por otros modelos, así como CMMI, metodologías ágiles, entre otros. Queda trabajo por hacer para llegar a niveles superiores de madurez que permitan tener un proceso refinado y productivo, y a su vez abordar los diferentes aspectos críticos del desarrollo de software, como la calidad, la resistencia al cambio y la escalabilidad. Del trabajo que se haga en este sentido, depende que tanto se aleje la industria de procesos artesanales que afectan su efectividad, oportunidad y eficiencia.

A partir de los resultados obtenidos con la realización de esta revisión sistemática, como trabajo futuro se espera abordar dos líneas de trabajo. En primer lugar, se espera llevar a cabo una segunda ejecución del protocolo en nuevas fuentes de información, esto con el objetivo de identificar un mayor número de trabajos que no fueron encontrados en las fuentes tratadas en esta revisión. En segundo lugar, diseñar una propuesta que permita conducir el desarrollo de software en mipymes a través de la integración de los dos enfoques analizados: (i) MDSD y (ii) enfoques ágiles. Con esta solución se espera poder facilitar el desarrollo de software basado en modelos y a través de un enfoque ágil en mipymes desarrolladoras de software.

Agradecimientos

César Pardo agradece la contribución de la Universidad del Cauca donde trabaja como profesor Asistente.

Referencias bibliográficas

Almeida, C. C. de J., & Oliveira, A. A. de. (2014). Qualitas: A Proposal of Process Model Development Software Driven Models. In *Proceedings of the 7th Euro American Conference on Telematics and Information Systems* (pp. 28:1–28:4). New York, NY, USA: ACM. doi:10.1145/2590651.2590678

Ambler, S. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process.* New York: John Wiley & Sons, Inc.

Ambler, S. (2004). *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press.

Andrade Almeida, J. P., Ferreira Pires, L., & Sinderen, M. J. van. (2004). Costs and benefits of multiple levels of models in MDA development. In *Second European Workshop on Model Driven Architecture with an Emphasis on Methodologies and Transformations* (pp. 12–20). Canterbury, UK.

Apps.co. (2016). *Impulso al desarrollo de aplicaciones móviles*. Recuperado Mayo 5, 2016, from https://apps.co/

Bancoldex. (2016). *Banco de desarrollo empresarial colombiano*. Recuperado Mayo 5, 2016, from https://goo.gl/FDHOJY

Batson, A., Lack, M., & Jones, A. (2015). *x86 Assembly Guide*. Recuperado de http://goo.gl/LMU44

Bernardo-Quintero, J., & Duitama-Muñoz, J. F. (2011). Reflexiones acerca de la adopción de enfoques centrados en modelos en el desarrollo de software. *Ingeniería Y Universidad*, *15*(1), 219–243.

Biolchini, J., Gomes, P., Cruz, A., & Travassos, G. (2005). *Systematic Review in Software Engineering*. Rio de Janeiro, Brazil: Systems

Engineering and Computer Science Department, UFRJ. Recuperado de http://goo.gl/ymUlwp

Brereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). Using a Protocol Template for Case Study Planning. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE 2008)* (pp. 1–8). Bari, Italy.

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Métodologías Ágiles en el Desarrollo de Software. In *VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2003)* (pp. 1–8). Alicante, España.

CaribeTIC. (2016). *Caribe en el mapa TIC global, las TIC en el mapa económico de la región*. Recuperado Mayo 5, 2016, from http://www.caribetic.com/

Chivers, I., & Sleightholme, J. (2016). *Introduction to Programming with Fortran*. London, UK: Springer London. doi:10.1007/978-0-85729-233-9

ClusterTIC. (2016). *Cluster tecnología, Información y Comunicación*. Recuperado de http://goo.gl/CzPrIC

Cockburn, A. (2006). *Cockburn: Agile Software Development: The cooperative Game*. Addison-Wesley Professional.

Duan, Y., & Fu, X. (2006). A Conceptual Approach to Modeling Model Driven Development Processes. In *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)* (pp. 179–198). Morne, Mauritius.

El Akkaoui, Z., Zimànyi, E., Mazón, J.-N., & Trujillo, J. (2011). A Model-driven Framework for ETL Process Development. In *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP* (pp. 45–52). New York, NY, USA: ACM.

Fallas, J. M. M. (2012). *Metodologías ágiles aplicadas a la Administración de Proyectos de Desarrollo de Software*. Valencia, España. Recuperado de http://goo.gl/EfQF4x

Fedesoft. (2016). *Federación Colombiana de la Industria de Software*. Recuperado de http://fedesoft.org/

Guta, G., Schreiner, W., & Draheim, D. (2009). A Lightweight MDSD Process Applied in Small Projects. In *35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009)* (pp. 255–258). Petras, Greece. doi:10.1109/SEAA.2009.63

Haase, A., Völter, M., Efftinge, S., & Kolb, B. (2007). Introduction to open Architecture Ware 4.1.2. In *Model-driven development tool implementers forum (MDD-TIF'07)*. Zurich, Switzerland.

Hebig, R., Gabrysiak, G., & Giese, H. (2012). Towards patterns for MDE-related processes to detect and handle changeability risks. In *International Conference on Software and System Process (ICSSP 2012)* (pp. 38–47). Zurich, Switzerland.

Hernández, M. G., León, B. B. P. de, & Ferrara, J. M. (2015). *MOSKITT, una herramienta libre que da soporte a la aplicación de metodologías de desarrollo*. Valencia. Recuperado de http://goo.gl/JF8SaI

INNPULSA. (2016). *Organismo para apoyar y promover el crecimiento empresarial extraordinario*. Recuperado de https://www.innpulsacolombia.com/

Intersoftware. (2016). *Federación Nacional de Software y Tecnologías de la Información*. Recuperado de http://goo.gl/UFrNHQ

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language (2nd edition)*. Prentice Hall.

Kirby, J. (2006). Model-Driven Agile Development of Reactive Multi-Agent Systems. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)* (Vol. 2, pp. 297–302).

Kraus, A., Knapp, A., & Koch, N. (2007). Model-Driven Generation of Web Applications in

UWE. In *3rd International Workshop on Model-Driven Web Engineering (MDWE 2007)*. Como, Italy.

Kroiss, C., Koch, N., & Knapp, A. (2009). WE4JSF - A Model-Driven Generation Approach for Web Applications. In M. Gaedke, M. Grossniklaus, & O. Díaz (Eds.), *LNCS: Web Engineering: 9th International Conference, ICWE 2009 San Sebastián, Spain, June 24-26, 2009 Proceedings* (pp. 493–496). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kroll, P., Kruchten, P., & Booch, G. (2003). *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP: A Practitioner's Guide to the RUP*. Addison-Wesley Professional.

Market, D. (2012). *Fedesoft impulsa la productividad en el sector del software y TI*. Recuperado de http://bit.ly/PnxtRo

Matinnejad, R. (2011). Agile Model Driven Development: An Intelligent Compromise. In *9th International Conference on Software Engineering Research, Management and Applications (SERA 2011)* (pp. 197–202). Baltimore, USA.

MinTIC. (2016). *Ministerio de Tecnologías de la Información y las Comunicaciones. Sector TIC*. Recuperado de http://www.mintic.gov.co/

Object Management Group. (2015). *Unified Modeling Language - UML 2.5*. Recuperado de http://www.omg.org/spec/UML/

Objects by Design. (2016). *UML products by product*. Recuperado de http://www.objectsbydesign.com/tools/umltools_byProduct.html

Oracle. (2015). *Oracle Data Werehousing Guide: Overview of Extraction, Transformation, and Loading*. Recuperado de http://goo.gl/brp9zB

PacifiTIC. (2016). *El Cluster TIC del Pacífico Colombiano*. Recuperado de http://goo.gl/EYKZCm

Paniza, J. (2011). *Aprende OpenXava con ejemplos*. CreateSpace Independent Publishing Platform.

Parquesoft. (2016). *Página oficial*. Recuperado de http://parquesoft.com/

RED CLUSTER COLOMBIA. (2016). *Red de clusters en Colombia*. Recuperado de http://redclustercolombia.com/

Rigworo, V. K. (2013). Adaptable model-driven engineering for formal methods integration with agile techniques for design of software systems. *Academic Research International*, *4*(1), 446–456.

Ruta N. (2016). *Página oficial*. Recuperado de http://www.rutanmedellin.org/

Sarin, A. (2011). *Spring Roo 1.1 Cookbook*. Packt Publishing.

SENA. (2016). *Servicio Nacional de Aprendizaje. Portal de oferta educativa*. Recuperado de http://oferta.senasofiaplus.edu.co/sofia-oferta/inicio-sofia-plus.html

SinerTIC. (2016). *Asociación Alianza Sinertic*. Recuperado de http://sinertic.org/

Terren, J. R., Moreno, D. A., & Jiménez, J. B. (2007). *Programacion Orientada a Objetos*. Marcombo S.A.

The Eclipse Foundation. (2012). *Model to Model Transformation – MMT*. Recuperado de https://goo.gl/s8H5Xk

The Eclipse Foundation. (2016). *Eclipse Modeling Framework (EMF)*. Recuperado de http://www.eclipse.org/modeling/emf/

Völter, M., Stahl, T., Bettin, J., Haase, A., Helsen, S., & Czarnecki, K. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons.

Zhang, Y., & Patel, S. (2011). Agile Model-Driven Development in Practice. *IEEE Software*, *28*(2), 84–91.