CMMI-DEV en el desarrollo de software bajo el paradigma de Desarrollo Guiado por Modelos, MDD*[1]

[CMMI-DEV in software development under Model-Driven software Development, MDD]

[CMMI-DEV no desenvolvimento de software sob o paradigma de Desenvolvimento Model-Driven, MDD]

Rosalba MATOS[2], Claudia PONS[3]

Recibo: 10.02.2016 – Aprobación: 30.09.2016

Resumen: *Continuando con la tarea de buscar nuevos modelos de desarrollo que mejoren el proceso de construcción de software, surgió en Gran Bretaña en 1994, el concepto de Desarrollo Guiado por Modelos, MDD, con tres objetivos principales: Aumentar la portabilidad, la interoperabilidad y la reutilización del software. En la actualidad MDD está ganando atención de la industria y las comunidades de investigación. Ya se encuentran casos de éxito documentados que declaran un ahorro de tiempo significativo en proyectos de desarrollo bajo MDD, donde la Arquitectura dirigida por modelos (MDA), es uno de los modelos más conocido, desarrollada por Object Management Group (OMG). Este trabajo tiene como objetivo presentar para cada etapa del proceso de desarrollo bajo MDA, cuáles son los artefactos que deben tenerse en cuenta para*

* **Modelo para la citación de este artículo:**
MATOS, Rosalba & PONS, Claudia (2016). CMMI-DEV en el desarrollo de software bajo el paradigma de Desarrollo Guiado por Modelos, MDD. En: Ventana Informática No. 35 (jul-dic). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 13-30. ISSN: 0123-9678

1 Artículo de investigación proveniente de la tesis *Cómo Integrar el Modelo CMMI al Modelo de Desarrollo de Software MDD*, para optar al título de Master en Ingeniería de software, por la Universidad Nacional de La Plata, durante el período *2013-2015*, por parte de la primera autora, bajo la dirección de la segunda.

2 Ing. de Sistemas; Esp. en Ingeniería de Software. Docente, Universidad Libre de Barranquilla (Barranquilla, Colombia). Correo electrónico: rosalba_matos@hotmail.com, rmatosm@unilibrebaq.edu.co

3 Licenciada en Informática, Especialista en docencia universitaria, Doctora en Ciencias Informáticas. Investigadora de la Comisión de Investigaciones Científicas CIC, Directora del Centro de Altos Estudios en Tecnología Informática (CAETI), Universidad Abierta Interamericana (Buenos Aires, Argentina). Correo electrónico: claudia.pons@uai.edu.ar

que un proyecto logre soportar 100% las prácticas específicas definidas por Capability Maturity Model Integration for Development (CMMI DEV 1.3), Niveles 2 y 3. Para ello se analizaron cada uno de los riesgos presentes en un desarrollo bajo MDD, y se establecieron recomendaciones que fueron validadas por varios investigadores sobre el tema de aplicación de los dos modelos CMMI y MDD en un mismo proyecto de desarrollo de software.

Palabras clave: *MDD, CMMI, MDA, OMG*

Abstract: *Continuing with the task of seeking new development models that improve the process of building software, the concept of Model-Driven Development, MDD emerged in Britain in 1994, with three main objectives: increasing portability, interoperability and reuse of software. Currently, the interest of industry and research communities on MDD is increasing so rapidly that it is already possible to find successful documented cases of successful declaring significant time savings in development projects under MDD. One of the best known MDD models is the MDA (Model Driven Architecture), developed by the Object Management Group - OMG. This work aims at presenting which artifacts are the most suitable to be considered in each stage of the development process under MDA, for a project to meet the specific practices defined by DEV CMMI 1.3, Levels 2 and 3. For this, each of the risks involved in a development under MDD were analyzed. Recommendations on the issue of implementation of both CMMI and MDD models in a software development project were defined and then validated with several researchers.*

Keywords*: MDD, CMMI, MDA, OMG.*

Resumo: *Continuando com a tarefa de buscar novos modelos de desenvolvimento que melhoram o processo de construção de software, surgiu na Grã-Bretanha em 1994, o conceito de Modelos de Desenvolvimento, MDD, com três objetivos principais: aumentar a portabilidade, interoperabilidade e reutilização software. MDD está ganhando a atenção das comunidades industriais e de investigação. Já documentados casos de sucesso de declarar uma significativa economia de tempo em projetos de desenvolvimento sob MDD, onde o Modelo Driven Architecture (MDA), é um dos modelos mais conhecidos, desenvolvido pelo Object Management Group (OMG). Este trabalho tem como objetivo apresentar para cada fase do processo de desenvolvimento sob MDA, os artefatos a ser tidos em conta são para um projeto para atingir 100% de apoio práticas específicas definidas por Capability Maturity Model*

Integration para o Desenvolvimento (CMMI DEV 1.3) os níveis 2 e 3. Foram analisados cada um dos riscos em desenvolvimento no âmbito MDD, e recomendações foram validados por diversos pesquisadores sobre o tema da aplicação de ambos os modelos CMMI e TDM no mesmo projeto de desenvolvimento estabelecido software.

Palavras-chave: *MDD, CMMI, MDA, OMG.*

Introducción

El avance tecnológico y el aumento de la complejidad de los sistemas han traído nuevos problemas para resolver a la Ingeniería de Software. Esto ha conducido a los investigadores a pensar en nuevas metodologías, herramientas y lenguajes que permitan realizar los procesos de la Ingeniería de Software de forma eficiente. El uso de modelos en el desarrollo de software surge como respuesta a esta necesidad, buscando lograr al final la generación automática del código, siendo el Desarrollo Guiado por Modelos (MDD), un ejemplo.

Debido a que la calidad del software depende de la calidad de los procesos de desarrollo, unir los beneficios de usar el paradigma de desarrollo dirigido por modelos a un modelo guía de mejora de los procesos de desarrollo, resulta muy prometedor. A pesar de esto, actualmente es poca la información que se encuentra sobre qué pautas tener en cuenta en un desarrollo MDD, para asegurar el éxito en las evaluaciones CMMI-DEV; aunque puede decirse que es posible encontrar mucha documentación de los modelos por separado.

Fueron varios los autores que contribuyeron con los resultados de sus investigaciones a reafirmar supuestos e identificar los artefactos que podrían aplicarse en un desarrollo MDD para cumplir con las guías CMMI-DEV. Entre ellos, Esterkin (2014, 100), quien al realizar el mapeo entre los dos modelos encontró que hay varias áreas de procesos y prácticas específicas de CMMI que no son soportadas por MDD.

Por otro lado, Quintero & Duitama (2014, 2) plantean los retos que aun afronta el MDD, como son las limitaciones de las herramientas y la falta de integración del modelo BPM en las transformaciones de modelos. Además, presentan las posibles razones para que puede fracasar un proyecto de desarrollo dirigido por modelos y proponen estrategias para mitigar ese riesgo, así como las características deseables en las herramientas, lenguajes y técnicas que soportan el desarrollo MDD.

Reforzando la justificación de incrementar y mejorar el uso de MDD, Martínez, Cachero & Meilá (2013, 189), concluyeron luego de realizar un experimento con 26 estudiantes de la Universidad de Alicante, que el paradigma de desarrollo dirigido por modelos es uno de los más difícil de entender, pero es el más útil en el largo plazo al compararlo con el modelo basado en código, y con el paradigma basado en modelos. De hecho, el experimento mostró cómo 20 de los 26 desarrolladores eligió MDD para continuar con el desarrollo del proyecto del experimento. Otro caso de estudio, es presentado por Teixeira & Dias (2015), con el objetivo de mostrar cómo se puede incrementar la productividad en el desarrollo de sistemas de software a través de la MDA, señalando la existencia de herramientas gratuitas que permiten desarrollar a bajo costo un sistema bajo el paradigma MDD.

El objetivo principal de este trabajo es brindar una guía al equipo de desarrollo de software bajo MDD, que le permita conocer qué artefactos debería implementar, si desea que sus procesos soporten las prácticas específicas definidas en CMMI DEV V.1.3, en los niveles 2 y 3.

El artículo está organizado de la siguiente forma: en la sección uno, se explican los principales conceptos teóricos necesarios para comprender la investigación, como son los procesos que se siguen en un desarrollo de software dirigido por modelos. En la segunda, se presenta la metodología utilizada en la investigación; en la sección tres se muestran y discuten los resultados obtenidos y, finalmente, las conclusiones y la bibliografía.

1. Fundamento teórico

MDD es un paradigma de desarrollo de software que utiliza modelos y cuyo objetivo es separar el diseño del sistema de la arquitectura de las tecnologías, para que puedan ser modificados independientemente. Busca tres objetivos principales: Portabilidad, la Interoperabilidad y la Reutilización que eventualmente deberían conducir a un aumento en la productividad. MDD se apoya en los modelos, las transformaciones entre modelos y los lenguajes específicos de dominio.

1.1 Proceso de desarrollo MDA (Arquitectura dirigida por modelos)

MDA surge aproximadamente en el año 2000 y, de acuerdo con Berre & Elvesæter (2008, 11), en el 2004 se acordó definirla durante una sesión plenaria del *Object and Reference Model Subcommittee*

of the Architecture Board (ORMSC), celebrada en Montreal: «*MDA es una iniciativa OMG que propone definir un conjunto de normas no patentadas que especificarán tecnologías interoperables con los que cuenta el desarrollo dirigido por modelos con transformaciones automatizadas*».

Es tanta la aceptación de MDA[4] que, según OMG (2015), apoya la evolución de las normas en dominios de aplicación tan diversos como la planificación de recursos empresariales, el control del tráfico aéreo y la investigación del genoma humano. Los procesos y actores principales de un proyecto de desarrollo (Tabla 1) bajo el paradigma MDA, definidas por Musat (2009, 20), son:

- El CIM: es el modelo más abstracto de todos los usados en MDA, y describe la lógica del dominio del negocio desde una perspectiva independiente de la computación.
- El PIM: es el segundo modelo en jerarquía de MDA. Describe todos los aspectos funcionales de la aplicación independientemente de la plataforma de software en la que se vaya a implementar y ejecutar.
- El PSM: es un modelo del sistema con detalles específicos de la plataforma en la que será implementado. Se genera a partir del PIM, así que representa el mismo sistema, pero con distinto nivel de abstracción.

Tabla 1. Actividades vs. Actores (Construido a partir de Mellor & Watson, 2004)

Actividad	ID del actor*						
	1	2	3	4	5	6	7
Identificar elementos del contexto del problema	X	X	X				
Seleccionar modelos		X	X	X			X
Afinar requerimientos			X	X			
Construcción y prueba del Modelo de Requerimientos	X		X	X			X
Transformar CIM en PIM y verificar completitud		X	X		X		X
Analizar la arquitectura			X	X			
Generar modelos PIM			X		X		X
Generar los modelos de diseño PSM´s		X	X	X	X		X
Generar el código básico	X	X	X			X	X

* Patrocinador, stakeholders, usuarios y clientes, (2) Líder del proyecto, (3) Arquitecto, (4) Analista de requerimientos, (5) Analistas diseñadores, (6) Analista / programadores, (7) Probadores y/o Auditores de Sistemas

4 Aborda el ciclo de vida completo de diseñar, desplegar, integrar y gestionar aplicaciones, así como el manejo de los datos usando estándares abiertos.

Las transformaciones para pasar de un modelo a otro, se constituyen en los pilares de MDA. Por ello, las herramientas de transformación de modelos juegan un papel importante en el desarrollo, y vale la pena revisar las características mínimas que se espera deben cumplir las herramientas de transformación de modelos para administrar los riesgos de las transformaciones.

Es recomendable que las herramientas de construcción y transformación de modelos cumplan las características de permitir a los desarrolladores ajustar las transformaciones y conocer el origen de cualquier elemento del modelo: trazabilidad, así como que los cambios hechos en el modelo inicial se conserven para cuando sea necesario volver a generar el modelo (consistencia incremental).

1.1.1 Generación del CIM. La figura 1 resume las actividades, secuencia y actores propios de la generación de un modelo independiente de la computación. Es pertinente considerar algunos de los riesgos que se deben administrar al elaborar un CIM:

- Que el modelo CIM tenga bajo nivel de exactitud.
- Que no contenga todas las reglas del negocio requeridas para el modelo del proceso de negocio que se modela.
- Según Quelopana, Vega & Meneses (2009, 5) es un riesgo que no se incluyan completamente todos los actores en el modelo, o sea, los primarios o tomadores de decisiones, los secundarios, o sea los que participan en la toma de decisiones, y los terciarios, que son los que producto del proceso de toma de decisiones, llevan a cabo las actividades o subprocesos operacionales.

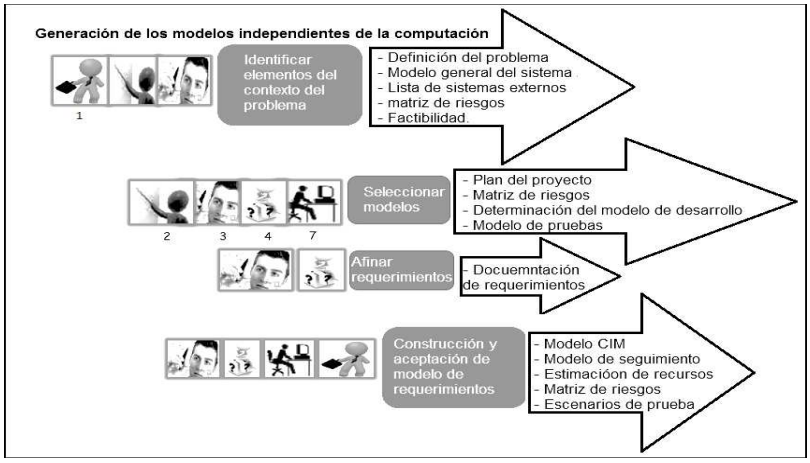

Figura 1. Actividades, y roles que intervienen en la generación del CIM

1.1.2 Generación del PIM. El PIM debe ser conductualmente completo (Figura 2), definiendo la lógica de negocio en términos de acciones abstractas. «*Lastimosamente, el proceso de desarrollo usando MDA, no considera la posibilidad de transformar el CIM en un PIM. Y aunque el CIM es bastante abstracto, parece necesario establecer un procedimiento que permita construir el PIM a partir del CIM, con la ayuda de una herramienta MDD y un conjunto de reglas de transformación*» (Chaparro & Gómez, 2012, 9).

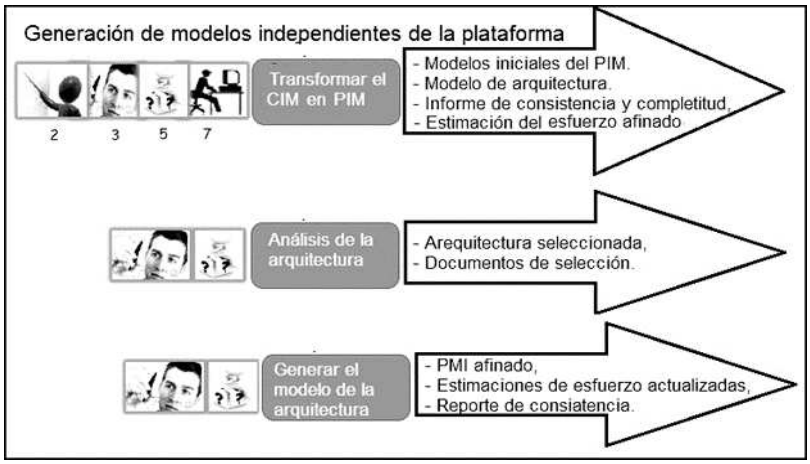

Figura 2. Actividades, y roles que intervienen en la generación del PIM

1.1.3 Generación del PSM. La figura 3a resume las actividades, secuencia y actores propios de la generación de un modelo específico de la Plataforma. Debido a que un PSM, es demasiado abstracto para la compilación en un idioma determinado, un entorno MDA requiere de un conjunto de reglas de transformación y una herramienta de Desarrollo de software guiado por modelos (MDD), para generar el código a partir del PSM. También puede ser necesario que se construyan varios PSM para los diferentes módulos del sistema, en cuyo caso también sería necesario utilizar Puentes, o herramientas de transformación, para que se comuniquen los diferentes PSM.

Los riesgos en esta etapa están inmersos en las siguientes actividades:

- Ejecutar reglas de transformación, - Verificar consistencia y completitud del modelo y - Verificar escenarios de prueba. Un riesgo que actualmente presentan las herramientas basadas en MDA es que no son fáciles de configurar inicialmente, ni de definir las transformaciones para una plataforma. Sin embargo, este esfuerzo es compensado cuando se utilizan estas transformaciones para múltiples aplicaciones.

Según Meaurio & Schmieder (2013, 145) «*MDA al permitir la doble transformación automática PIMPSM y PSM-código fuente hace que el esfuerzo se centre en los modelos de alto nivel, y al regenerar el resto de los modelos, la trazabilidad está garantizada. Esto hace que las iteraciones dentro del ciclo de vida sean más eficaces para afrontar estos cambios, minimizando de esta manera los riegos asociados*».

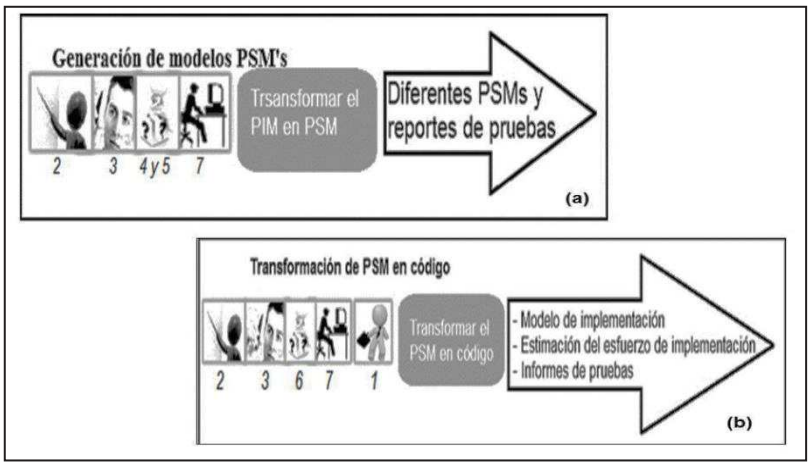

Figura 3. Actividades y roles

1.1.4 Generación del Código. La figura 3b resume las actividades, secuencia y actores propios de la generación del código. En esta actividad el Arquitecto, y el desarrollador utilizan las herramientas y reglas de transformación para generar el código básico de la aplicación que se desarrolla. Los Programadores al trabajar con herramientas MDA descubren que pueden escribir el código como parte de una plantilla de generación de código, dándoles más tiempo para trabajar en las piezas individuales más interesantes de la lógica que el código de un generador de código. La generación de código se puede hacer para varios lenguajes de programación como Java, C, Visual Basic.NET y C ++ entre otros.

1.2 CMMI (Capability Maturity Model Integration for Development)

CMMI-DEV «*proporciona una orientación para aplicar las buenas prácticas CMMI en una organización de desarrollo. Las buenas prácticas del modelo se centran en las actividades para desarrollar productos y servicios de calidad con el fin de cumplir las necesidades de clientes y usuarios finales*» (Chrissis, Konrad & Shrum, 2012). Con base en Puello (2013, 21-22), se hace un resumen histórico de CMMI:

- Sus principios se remontan a 1984, cuando en Estados Unidos se aprobó la creación de un organismo de investigación (SEI, Instituto de Ingeniería de Software) para la mejora en el desarrollo de los sistemas de software y evaluación de respuesta y fiabilidad de las compañías que suministran software al Departamento de Defensa.
- En 1985 el SEI origina el Modelo de Madurez de las Capacidades (CMM) y en 1991 publica la versión 1.0 del Modelo de Madurez de las Capacidades para el Software (SW-CMM, *Capability Maturity Model for Software*).
- A mediados de la década del 90, el SEI unifica los modelos de ingeniería de software (SW-CMM), de ingeniería de sistemas (SE-CMM) y de desarrollo integrado de productos (IPD-CMM), emergiendo una nueva generación llamada CMMISM (*Capability Maturity Model Integration*), que libera la versión 1.1 en 2002, que sirve de guía para mejorar los procesos organizacionales, además del desarrollo del Software.

1.3 Marco referencial

Algunos autores han abordado el tema de la utilización del modelo de madurez de capacidades CMMI-DEV en proyectos de desarrollo utilizando el paradigma de desarrollo dirigido por modelos:

- Esterkin (2014), presenta los resultados de un análisis para responder a la pregunta si las prácticas MDD, soportan el nivel de madurez 2 del CMMI. Como resultado del mapeo encuentra que hay varias áreas de procesos y prácticas específicas no soportadas por MDD.
- Esterkin & Pons (2012, 736-741), realizan un análisis de las buenas prácticas de MDD y determinan si éstas soportan las guías del nivel 2 de CMMI, y en qué medida las soportan.
- SEI (2010, 58), explica cómo interpretar las directrices del modelo CMMI para ser usadas en proyectos de desarrollo relacionadas con las Metodologías ágiles[5].
- En la búsqueda de una forma de evaluar la calidad en un proyecto de MDD, Ríos et al., (2006), muestran cómo un modelo dentro del proyecto *Modelware*[6] ayudó a varias compañías para adoptar el modelo de desarrollo MDD.

5 Esta posibilidad de usar el modelo de referencia CMMI y una metodología particular de desarrollo, ha servido de base para que Inigo Garro, consultor de CMMI, plantee cuáles son los aspectos claves que permiten la coexistencia de CMMI y los enfoques ágiles.

6 El proyecto *Modelware* fue desarrollado en España entre 2002 y 2006, definiendo cinco niveles de madurez que ubicaban a las empresas que adoptaban MDD en diferentes grados tanto de seguimiento de prácticas como de uso de artefactos.

2. Metodología

Esta investigación fue abordada desde un análisis de los riesgos de cada uno de los procesos en un desarrollo de software utilizando MDA, por ser uno de los modelos MDD más conocidos.

Luego de realizar las consultas pertinentes para conocer el estado del arte de ambos modelos, CMMI y MDD; confirmar la necesidad de vincular temas de mejora de procesos al paradigma de desarrollo MDD, y comprobar la escasez de trabajos documentados relacionados con el tema, se determinó que debía realizarse un análisis de los riesgos en cada una de las actividades de las etapas de desarrollo bajo MDA. Específicamente se analizaron los riesgos propios del modelo MDA, y luego se mapearon las actividades con riesgos, a las practicas específicas de las áreas de procesos CMMI-DEV 1.3

Por ser MDA un modelo de desarrollo, se limitaron las áreas de proceso que debían ser analizadas a las áreas de proceso que se centran en las prácticas específicas del desarrollo: Planificación del proyecto, Seguimiento y control del proyecto, Gestión de requerimientos, Gestión de la configuración, y Aseguramiento de la calidad del proceso y el producto, para el nivel 2, excluyendo Gestión de acuerdos con proveedores y Medición y análisis, que pueden llevarse de manera similar, independientemente de la metodología de desarrollo. Por la misma razón se incluyeron solo las áreas de Integración de Producto, Desarrollo de Requerimientos, Gestión de Riesgos, Solución Técnica, Integración del producto, Validación, Verificación, y Gestión integrada del proyecto para el nivel 3, y no se consideraron áreas para los niveles 4 y 5.

En la tarea de buscar los riesgos, se estableció como criterio de selección aquellos con alta probabilidad, debido a particularidades del MDD. Para ello, se analizaron para cada una de las etapas de desarrollo con MDA, el modelo correspondiente en cada etapa, las herramientas usadas, la transformación al siguiente modelo, y los riesgos existentes en cada transformación. Posteriormente se investigó sobre las actividades de cada etapa, las personas involucradas y por último de determinaron los controles que deberían implementarse para prevenir o minimizar los riesgos anteriormente identificados.

2.1 Validación de la propuesta

Con el fin de validar la propuesta, se realizó una encuesta[7] difundida en dos foros de CMMI, y un foro sobre MDD y se invitó por correo

7 La encuesta está disponible en: http://goo.gl/apcl50 , mientras sus resultados en goo.gl/vtSo2P

electrónico a algunos autores de artículos, libros y tesis citados en este trabajo, para que dieran su opinión sobre la propuesta. En busca de lograr la comprensión de la propuesta por parte de los colaboradores, se publicó un artículo previo, donde se presentó la propuesta.

3. Resultados y discusión

3.1 Descripción de resultados

Con base en los riesgos identificados, y tomando los resultados del análisis de Esterkin (2014, 80), sobre las prácticas específicas definidas por CMMI DEV 1.3 Nivel 2 soportadas en MDD, se describe el problema, y se identifican y proponen los artefactos que podrían utilizarse para satisfacer las prácticas CMMI que podrían no cumplirse. Además, se complementan las recomendaciones para soportar también CMMI DEV 1.3 Nivel 3. De esta forma se responde en parte la pregunta sobre como cumplir con CMMI-DEV 1.3 nivel 2 y 3, en un proyecto MDD. En la Tabla 2 se presenta el resumen de los resultados obtenidos en el proyecto[8].

Por otro lado, se encuentran algunos riesgos que se gestionan con la utilización de artefactos no propios de MDD, pero que deben ser especialmente atendidos por la misma naturaleza del desarrollo bajo MDD. Por ejemplo, es necesario establecer un marco de gestión de la configuración que ayude a controlar la transformación de modelos y facilite la verificación y la validación de la exactitud de los requerimientos. Por eso se propone utilizar los modelos de trazado a partir de metamodelos, o modelos de lenguajes de modelado, que facilitan la utilización de patrones dinámicos que estandarizan y guían la práctica de la trazabilidad a partir de los criterios de calidad. Es decir, ir más allá del uso de plantillas para formalizar las solicitudes de cambio. En cuanto a la matriz de riesgos, sería conveniente elaborarla teniendo en cuenta el registro de riesgos por actividades propias del desarrollo MDD.

8 Vale la pena aclarar que en esta tabla solo se mencionan las áreas de proceso y prácticas para las que se identificaron los riesgos altos de incumplimiento debido a las particularidades del desarrollo bajo MDD, que podrían ser mitigados con algún artefacto.

Tabla 2.

Resumen de los artefactos propuestos para cumplir con prácticas específicas de CMMI v1.3 niveles 2 y 3

Nivel	Área de proceso/ SP con alto riesgo propio en MDD	Problema	Artefacto propuesto
2	Gestión de requerimientos (REQM). - SP 1.4 Mantener una trazabilidad bidireccional de los requerimientos	«*Puede verse afectada en MDD por factores como la complejidad de los modelos, cambios invasivos no controlados, modelos parcialmente actualizados cuando hay cambios, matrices de trazabilidad no actualizadas, y altos costo para la realización de esta práctica*» (Tabares, 2009, 7).	Herramientas de gestión del trabajo y de configuración. MDD requiere de herramientas que van más allá de las matrices de trazabilidad[1]* de requerimientos. Utilizar los modelos de trazado a partir de metamodelos, o modelos de lenguajes de modelado, facilita la utilización de patrones dinámicos que estandarizan y guían la práctica de la trazabilidad a partir de los criterios de calidad.
	Seguimiento y Control de Proyecto (PMC). Se tienen cinco prácticas específicas no soportadas.	Que no se incluyan completamente todos los actores en el modelo. Cada actor realiza actividades críticas, puntuales y diferentes que difícilmente podrían ser ejecutadas por otro actor.	Especial cuidado en la asignación y control de participación de cada rol en las etapas del desarrollo (Tabla 1). Como las prácticas se pueden conducir en MDD, de manera análoga a un desarrollo tradicional, se analiza la práctica que merece tratarse diferente según SP 1.5 (Monitorear la Participación de los *stakeholders*)
	Planeamiento del Proyecto (PP). - SP 3.1 Revisar los planes que afectan el proyecto. - SP 3.2 Conciliar los niveles de trabajo y de recursos.	MDD no tiene prácticas propias que soporten las prácticas específicas 3.1 y 3.2, Además, en los desarrollos dirigidos por modelos es especialmente importante planificar el alcance, la estimación de recursos y la matriz de riesgos.	Artefactos usados en un desarrollo tradicional (registro de revisiones de planes que afectan el proyecto, negociación de recursos para conciliar lo planeado y disponible). El líder, el Arquitecto y el Analista de Requerimientos aseguran se incluyan las reglas del negocio requeridas, se verifique y valide por el probador y los usuarios, usan técnicas y tecnologías de soporte de la construcción sin ambigüedades[2]*.
	Gestión de la Configuración (CM). - SP 2.1 Rastrear los pedidos de cambio.	En el desarrollo dirigido por modelos es importante establecer un marco de gestión de la configuración que ayude a controlar la transformación de modelos y facilite la verificación y la validación de la exactitud de los requerimientos.	Se hace necesario fortalecer las prácticas de Gestión de la configuración y utilizar más que plantillas para formalizar las solicitudes de cambio. Utilizar los modelos de trazado a partir de metamodelos, o modelos de lenguajes de modelado, facilita la utilización de patrones dinámicos que estandarizan y guían la práctica de la trazabilidad[3]* a partir de los criterios de calidad.

	Aseguramiento de la Calidad del Proceso y del Producto (PPQA). - SP 2.1 Comunicar y solucionar problemas no resueltos.	«*Falta de procedimientos que indiquen la necesidad de registrar los problemas de incumplimiento*» (Esterkin, 2014, 97). La trazabilidad de requerimientos es un atributo de calidad tratado por la ingeniería de software, y el desarrollo por Modelos, requiere control de su consistencia y completitud.	Se deberían implementar reportes de acciones correctivas y reportes de evaluación. El modelo de trazado asumido en el proyecto de desarrollo y definido en el Plan de Calidad de la empresa son herramientas de Aseguramiento de la Calidad del producto y del proceso
3	Integración de Producto (PI). - SP 3.2 Ensamblar los componentes de producto. - SP 3.3 Evaluar los componentes de producto ensamblados.	El producto es construido a partir de componentes heterogéneos, ignorando las diferencias de implementación entre los componentes o aplicaciones. Así MDD atiende por naturaleza el asunto de la integración, sin descuidar el cumplimiento de estándares de los componentes.	Se recomienda tener un mecanismo para certificar que los artefactos y modelos cumplan los estándares y se mantenga la integridad del sistema
	Desarrollo de Requerimientos (RD). - SP 1.2 trasformar las necesidades de las partes interesadas en requisitos de cliente.	En un desarrollo MDD, cuando el modelado inicia en el nivel de requerimientos, se construye el Modelo Independiente de la Computación, CIM[4*], con las reglas de negocio y los requerimientos del sistema. Lo ideal es disponer de herramientas que permitan la transformación CIM a PIM, aunque se dificulte porque el CIM es bastante abstracto.	Se recomienda utilizar herramientas que faciliten la transformación automática de CIM a PIM. En los casos en que el equipo decida iniciar la construcción de modelos posterior a la etapa de refinamiento de requerimientos, el equipo deberá definir los artefactos que serán los ejes de trazado de requerimientos. «*Este procedimiento podría utilizar una herramienta MDD y un conjunto de reglas de transformación*» (Chaparro & Gómez, 2012, 77).
	Gestión de Riesgos (RSKM). - SP 1.3 establecer una estrategia de gestión de riesgos. - SP 2.1 Identificar los riesgos.	La gestión de riesgos es importante en MDD ya que cada construcción y transformación de modelos tiene riesgos que deben mitigarse en la construcción de un CIM y su transformación a PIM; Riesgos en la construcción de un PIM y su transformación a PSM; Riesgos en la construcción de un PSM y su transformación a código	Se recomienda elaborar la matriz de riesgos vs actividad de desarrollo. Esta matriz debería probarse y considerar los correctivos necesarios en las actividades de Construcción del modelo de requerimientos: Validaciones de completitud y consistencia, Transformación CIM a PIM, Generación de modelos de arquitectura, Transformación a código. En MDD, la transformación controlada por los modelos de trazado ayuda a disminuir los riesgos por subjetividad en las decisiones.
	Solución Técnica (TS). - SP 1.2 seleccionar las soluciones de componentes de producto.	Falta de definición de qué herramientas utilizar y qué modelos reutilizar.	Según Swithinbank et al. (2005, 66), el arquitecto de soluciones deberá elegir el tipo de modelo apropiado, (UML u otro), para que utilicen los desarrolladores de la aplicación. También deben definir las herramientas MDD necesarias para desarrollar el proyecto.
	Verificación (VER) - SP 3.1 realizar la verificación.	Según Esterkin & Pons (2012, 4), es necesario realizar verificación tanto del framework del modelo como de todos los artefactos generados.	Verificar en cada actividad de transformación de modelos entre lo construido y los requerimientos, y cada herramienta desarrollada. Los casos de prueba deben registrar automáticamente sus resultados.

1 Tabares (2009, 7) describe un nuevo enfoque de trazabilidad (Modelos de trazado) como un patrón que controla la transformación de modelos y atiende el problema de actualización con las matrices de trazabilidad, mejorando la consistencia y completitud de los modelos que se transforman.
2 Por ejemplo, para la construcción del CIM es recomendable el uso de un lenguaje gráfico notacional que permita la construcción de modelos más precisos que los construidos usando UML.
3 Definir modelos de trazabilidad como controladores de la transformación de los modelos en el contexto MDD mejora la gestión de la configuración en servicios como: la verificación de la consistencia y la completitud, y el manejo del cambio (Tabares, 2009, 78).
4 Existen varios estándares y herramientas que pueden utilizarse para la construcción del CIM y que proporcionan una serie de elementos para representar de manera estandarizada los métodos, ciclos de vida, roles, actividades, tareas y productos de trabajo utilizados en Ingeniería de software.

3.2 Discusión de resultados

Los resultados del trabajo fueron compartidos con las personas que respondieron una encuesta cuyo objetivo fue conocer la opinión de los expertos en ambos modelos, acerca de si las prácticas que se realizan en un proyecto de desarrollo de software que utilice el paradigma MDD son suficientes para cumplir 100% con las prácticas específicas definidas en CMMI DEV 1.3, en los niveles 2 y 3, o si es necesario la implementación de diferentes artefactos o enriquecer los que se aplican en un desarrollo tradicional, para su logro.

La encuesta fue respondida, por 73 personas[9], entre mayo 02 y septiembre 16 de 2015, con la única dificultad de conseguir personas conocedoras del paradigma de desarrollo MDD, que se sintieran seguros de dar su opinión. El 37% (25 personas) afirmaron conocer los dos modelos MDD y CMMI, de las cuales un 24% declararon que las guías actuales de CMMI no son suficientes para guiar los procesos en un desarrollo bajo MDD y el 44% que no lo saben; 79% señalaron su interés por conocer cómo usar ambos modelos en un mismo proyecto de desarrollo. Además, un 68% afirmaron que los modelos CMMI y MDD deberían complementarse en un proyecto de desarrollo MDD.

En relación con los artefactos identificados para ser implementados en un desarrollo dirigido por modelos con el fin de soportar las guías de calidad CMMI DEV nivel 2 y 3, no se obtuvo ninguna retroalimentación. Sin embargo, Pat O'Toole, un instructor certificado de CMMI-DEV (http://cmmiinstitute.com/conferences/speakers/pat-otoole), hizo un aporte importante relacionado con la propuesta inicial de incluir notas aclaratorias al modelo CMMI-DEV que facilitaran el cumplimiento, y la verificación de estos artefactos en un desarrollo MDD, de manera análoga a cómo se interpretan metodologías ágiles con CMMI. Su

9 11 desarrolladores de software, seis directores de proyectos, cinco SCAMPI Lead Appraiser, cuatro profesores de universidades, cuatro consultores, tres investigadores, y otros.

concepto fue que la incorporación de los conceptos y notas MDD en el CMMI sería un trabajo mucho más extenso, y dado el nuevo enfoque del SEI en el proyecto de próxima generación, como lo confirma Basu (2015, 37), es improbable que en este momento se realicen tales cambios. Atendiendo esta alarma, se cambió la idea de sugerir la incorporación de notas, por la recomendación de que los artefactos sean considerados en una Process Asset Library (PAL), que implemente un proceso estándar para la organización, siguiendo los estándares de CMMI.

4. Conclusiones

En este trabajo se abordaron dos modelos importantes en el desarrollo de software: Uno con mucho reconocimiento y muchos casos de éxito documentados en el ámbito mundial, autoridad en el tema de la mejora y evaluación de los procesos de desarrollo de software, y otro, que, a pesar de no estar maduro, promete darle un gran impulso a la industria del software, y cambiar el nivel de los procesos de desarrollo.

Al finalizar la investigación y luego de validar la propuesta, se pudo concluir lo siguiente:

- Es necesario documentar en un proyecto de desarrollo bajo MDD, la manera cómo se podría evaluar el cumplimiento de las guías CMMI-DEV.
- Existen riesgos que podrían afectar la calidad del producto, en un proyecto de desarrollo de software bajo MDD, que no son mitigados si se cumplen sólo prácticas del paradigma MDD o del desarrollo tradicional.
- Es válida la propuesta de utilizar artefactos adicionales que permitan asegurar el cumplimiento de CMMI DEV en los niveles 2 y 3.
- Una buena forma de integrar los artefactos que permitan el cumplimento de prácticas específicas definidas por CMMI DEV 1.3 Nivel 2 y 3, sería considerarlas en una Process Asset Library, PAL.

Se justifica la continuación de estudios similares ya que los equipos de desarrollo de software que aplican MDD, necesitan de un estándar que defina los lineamientos y buenas prácticas del proceso de desarrollo, teniendo en cuenta los riesgos y particularidades del paradigma de desarrollo por modelos, y los evaluadores CMMI, debe cubrir las necesidades del sector que será cada vez creciente de equipos de desarrollo y empresas que decidan incorporar nuevos paradigmas de desarrollo de software.

Sin embargo, para que esa integración se dé, es necesario que los investigadores,

los fabricantes de herramientas MDD, la academia y el instituto CMMI, filial de Carnegie Innovations y quien es ahora responsable de todo lo relacionado con el entrenamiento, certificaciones, evaluaciones y mejora de procesos del modelo CMMI reúnan esfuerzos para interpretar las prácticas CMMI dentro de un proceso de desarrollo MDD. Esto incluiría por lo menos las siguientes actividades:

- Incorporar las notas aclaratorias necesarias en el modelo CMMI-DEV.
- Definir las prácticas a evaluar en cada proceso.
- Documentar y difundir las mejores prácticas como parte del modelo CMMI.
- Capacitar y certificar a los evaluadores de procesos de una empresa que desarrolle bajo MDD.

5. Agradecimientos

Se hace reconocimiento a la valiosa colaboración brindada por cada una de las personas que respondieron la encuesta de validación de la propuesta. Se agradece en especial a Pat O'Toole, quien generosamente realizó la revisión del artículo previo y de las preguntas de la encuesta realizada.

Referencias bibliográficas

BASU, Anirban (2015). Ware quality assurance, testing and metrics. New Delhi (India): Star Print-O-Bind. 301 p. ISBN: 978-81-203-5068-7.

BERRE, Arne-Jørgen & ELVESÆTER, Brian (2008). Model Based System Development: Part I – MDE, Model Driven Engineering [online]. In: BERRE, Arne-Jørgen. Course INF5120 - Modelbased System development: Lecture Notes for Course. Oslo (Norway): University of Oslo. 51 p. <http://www.uio.no/studier/emner/matnat/ifi/INF5120/v08/undervisningsmateriale/INF5120-Part-I-MDE.pdf> [consult: 15/04/2016]

CHAPARRO LEMUS, Luis Oliverio & GÓMEZ ESTUPIÑÁN, Juan Federico (2012). Una visión del desarrollo de software utilizando modelos [en línea]. En: Gerencia Tecnológica Informática, Vol. 11, No. 29. (Ene-Abr). Bucaramanga (Colombia): Universidad Industrial de Santander. p. 69-82. ISSN: 1657-8236 <http://revistas.uis.edu.co/index.php/revistagti/article/view/2818/3060> [consulta: 20/01/2016]

CHRISSIS, Mary Beth; KONRAD, Mike & SHRUM, Sandy (2012). CMMI para Desarrollo: Guía para la integración de procesos y la mejora de productos. Madrid (España): Editorial Universitaria Ramón Areces. 722 p. ISBN: 978-8499610788

ESTERKIN, Viviana & PONS, Claudia (2012). Análisis y Evaluación del MDD (Model Driven software Development) desde la Perspectiva del Nivel 2 del CMMI -DEV 1.3 [en línea]. En: XVIII Congreso Argentino de Ciencias de la Computación, CACID 2012 (08 -12/10/2012), Bahía Blanca (Buenos Aires, Argentina): Red de Universidades con Carreras en Informática, RedUNCI. Anales CACID 2012, p. 734-743. <http://sedici.unlp.edu.ar/handle/10915/23705>, <http://sedici.unlp.edu.ar/bitstream/handle/10915/23705/Documento_completo.pdf?sequence=1> [consulta: 14/03/2015]

ESTERKIN, Viviana (2014). Análisis y Evaluación del MDD (Model Driven Development) desde la perspectiva de CMMI DEV 1.3 Nivel 2. Tesis de grado (Magister en Tecnología Informática). Buenos Aires (Argentina): Universidad Abierta Interamericana, UAI. 161 p.

MARTÍNEZ ESPINOSA, Yulkeidi; CACHERO, Cristina & MELIÁ, Santiago (2013). MDD vs. traditional software development: A practitioner's subjective perspective [online]. In: Information and Software Technology, Vol. 55, No. 2 (feb). Newton (MA, USA): Butterworth-Heinemann. p. 189-200. ISSN: 0950-5849. <http://dx.doi.org/10.1016/j.infsof.2012.07.004>, <http://www.sciencedirect.com/science/article/pii/S0950584912001309> [consult: 12/09/2016]

MEAURIO, Valeria S. & SCHMIEDER, Eric (2013). La Arquitectura de Software en el Proceso de Desarrollo [en línea]. En: Revista Latinoamericana de Ingeniería de Software, Vol. 1, No. 4, Lanús (Buenos Aires, Argentina): Red de Ingeniería de Software de Latinoamérica, RedISLA. p. 142-146. ISSN: 2314-2642 <http://revistas.unla.edu.ar/software/article/download/103/77> [consulta: 10/02/2016].

MELLOR, Stephen J. & Watson Andrew (2004). Roles in the MDA Process [on line]. Alabama (USA): Object Management Group, OMG. <http://www.omg.org/registration/Roles_in_MDA1.pdf> [consult: 17/07/2016].

MUSAT SALVADOR, David (2009). Espora: Definición de lenguajes de operación específicos de dominios siguiendo un proceso de desarrollo dirigido por modelos [en línea]. Proyecto fin de carrera (Ingeniero Técnico en Informática de Sistemas). Madrid (España): Universidad Politécnica de Madrid, Escuela Universitaria de Informática, Departamento de Organización y Estructura de la Información. 102 p. <http://oa.upm.es/1392/2/PFC_DAVD_MUSAT_SALVADOR_DEFINITIVO.pdf> [consulta: 17/07/2016]

OBJECT MANAGEMENT GROUP, OMG (2015): Model Driven Architecture: The Architecture of Choice for a Changing World - Executive Overview [online]. Needham (MA, USA): OMG. (Last updated: 31/05/2015). <http://www.omg.org/mda/executive_overview.htm> [consult: 25/04/2015].

PUELLO, Oswaldo R. (2013). Modelo de verificación y validación basado en CMMI [en línea]. En: Investigación e Innovación en Ingenierías, Vol. 1, No. 1 (ene-jun). Barranquilla (Atlántico, Colombia): Universidad Simón Bolívar, Instituto de Investigaciones Científicas. p. 20-27. ISSN: 2344-8652 <http://publicaciones.unisimonbolivar.edu.co:82/rdigital/ojs/index.php/innovacioning/article/view/478/474> [consulta: 16/07/2016]

QUELOPANA, Aldo; VEGA Z., Vianca & MENESES V., Claudio (2009). Una propuesta metodológica para modelar procesos de negocio de decisión basada en una extensión a BPMN [en línea]. En: 3er Encuentro Informática y Gestión: Workshop Internacional EIG2009 (03-04/12/2009). Temuco (Chile): Universidad de La Frontera. CEUR Workshop Proceedings. ISBN: 978-956-236-204-7 <http://ceur-ws.org/Vol-558/Art_12.pdf> [consulta: 13/03/2016]

QUINTERO, Juan Bernardo & DUITAMA MUÑOZ, Jhon Freddy (2014). Reflexiones acerca de la adopción de enfoques centrados en modelos en el desarrollo de software [en línea]. En: Ingeniería y Universidad, Vol. 15, No. 1 (ene-jun). Bogotá (Colombia): Pontificia Universidad Javeriana. p. 219-243. ISSN: 0123-2126. <http://revistas.javeriana.edu.co/index.php/iyu/article/view/1131/810> [consulta: 12/02/2016]

RIOS, Erkuden; BOZHEVA, Teodora; BEDIAGA, Aitor & GUILLOREAU, Nathalie (2006). MDD Maturity Model: A Roadmap for Introducing Model-Driven Development. In: Second European Conference on Model Driven Architecture® Foundations and Applications, ECMDA-FA'06 (10-13/07/2006), Bilbao (Spain): ECMDA. RENSIK, Arend & WARMER, Jos (eds.). Model Driven Architecture – Foundations and Applications: Proceedings of the ECMDA-FA'06, Berlin (Germany): Springer. p. 78-89. ISBN: 3-540-35909-5

SOFWARE ENGINEERING INSTITUTE, SEI (2010). CMMI para desarrollo, Versión 1.3: CMMI-DEV, V1.3. Pittsburgh (USA): Carnegie Mellon University. 482 p. ISBN: 978-1-4467-5714-7.

SWITHINBANK, Peter; CHESSELL, Mandy; GARDNER, Tracy; GRIFFIN, Catherine; MAN, Jessica; WYLIE, Helen & YUSUF, Larry (2005). Patterns: Model-Driven Development Using IBM Rational Software Architect [Redbooks]. (USA). IBM, International Technical Support Organization. 236 p. ISBN: 9780738492889. <http://www.redbooks.ibm.com/redbooks/pdfs/sg247105.pdf> [consult: 13/10/2015]

TABARES BETANCUR, Marta Silvia (2009). Un patrón de trazabilidad para controlar la evolución de los intereses en un espacio multidimensional. Tesis doctoral (Doctor en Filosofía de Ingeniería -Sistemas) Medellín (Colombia): Universidad Nacional de Colombia, Escuela de Sistemas e Informática. 272 p.

TEIXEIRA COSTA, Pedro Henrique & DIAS CANEDO, Edna (2015). Using a MDD approach to develop software systems [online]. In: 10th Iberian Conference on Information Systems and Technologies, CISTI 2015 (17-20/06/2015). Aveiro (Portugal): AISTI, Associação Ibérica de Sistemas e Tecnologias de Informação. Proceedings of the CISTI 2015. Washington, D.C. (USA): Institute of Electrical and Electronics Engineers, IEEE. p. 108-113. e-ISBN: 978-9-8998-4345-5. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7170371>, <DOI: 10.1109/CISTI.2015.7170371> [consult: 01/10/2016]