Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 156-171
http://rcci.uci.cu

Tipo de artículo: Artículo de revisión
Temática: Desarrollo de aplicaciones informáticas
Recibido: 23/10/2013 | Aceptado: 18/11/2013 | Publicado: 21/01/2014

Estado de la complejidad arbitraria y Arquitectura Dirigida por Modelos en el desarrollo de software en Cuba

State of arbitrary complexity and Model Driven Architecture in the software development in Cuba

Nemury Silega [1]*, Danaysa Macías [2]*, Juan Pedro Febles[3], Manuel Noguera[4]

[1] CEIGE. Centro de Gestión de Entidades. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños km 2 ½, Reparto Torrens, Boyeros, La Habana, Cuba. CP.: 19370
[2] CEGEL. Centro de Gobierno Electrónico. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños km 2 ½, Reparto Torrens, Boyeros, La Habana, Cuba. CP.: 19370
[3] Centro Internacional del Postgrado. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP.: 19370
[4] Departamento de Ingeniería de Software. Universidad de Granada, E.T.S.I.I., c/Saucedo Aranda s/n, 18071 Granada, España

* Autor para correspondencia: dmacias@uci.cu
{nsilega, febles}@uci.cu; mnoguera@ugr.es

Resumen

Las aplicaciones informáticas tienen un papel determinante en la sociedad actual, ya que prácticamente en todas las esferas de la vida se utiliza algún software. La comunidad de desarrollo e investigación de software dedica sustanciales esfuerzos a determinar cómo los sistemas de software pueden mejorar cada vez más el entorno donde son

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 156-171
http://rcci.uci.cu

aplicados, como por ejemplo en la medicina, la gestión empresarial, la educación, entre otros dominios. Sin embargo y paradójicamente estos esfuerzos no se corresponden con los que se dedican a mejorar el entorno propio del desarrollo de software. Todavía resultan insuficientes las propuestas encaminadas a agilizar el proceso de desarrollo de software. Los autores realizan un análisis de varias propuestas encaminadas a mejorar el proceso de desarrollo de software en Cuba, proponen cómo abordar el problema sobre la base del paradigma de Arquitectura Dirigida por Modelos y hacen una valoración del impacto que esta puede tener para la industria cubana de software.
Palabras clave: Desarrollo de software, complejidad arbitraria, arquitectura dirigida por modelos.

Abstract

Informatic systems play an important role in the society because are used in the majority of fields. The community of software researchers and developers devotes substantial efforts to determine how software systems can increasingly improve the environment in which they are deployed, for example, in medicine, business management, education, among other domains. But paradoxically these efforts do not correspond to those devoted to improving the environment of software development itself. There are insufficient proposals to streamline the software development process. The authors conducted an analysis of several proposals to improve the software development process in Cuba, propose how to address the problem based on the paradigm of Model Driven Architecture and make an assessment of the impact this may have on Cuban software industry.
Keywords: *Arbitrary complexity, model driven architecture, software development.*

Introducción

Desde prácticamente los comienzos de la ingeniería del software, diversos investigadores han reconocido la relevancia de llevar a cabo investigaciones con el objetivo de mejorar el proceso propio de desarrollo de software, como apuntaba (Naur, 1968) *"Un objetivo importante de la ingeniería de software es permitir a los desarrolladores construir sistemas que operen de manera confiable proporcionando teorías, métodos y herramientas para el desarrollo de software de calidad"*. Sin embargo, los mayores esfuerzos de la comunidad de desarrollo e investigadores de software se han dedicado a mejorar otras áreas de la sociedad, paradójicamente olvidando un tanto su propia área.

Este tema, el reconocido investigador Frederick Brooks, lo analizó en (Brooks, 1995) y concluyó que el proceso de software era un proceso complejo. Esta complejidad tiene dos dimensiones. La primera de ellas está enmarcada en el entorno donde se intenta mejorar con el software, esta complejidad Brooks la denominó *esencial*, porque es inherente al entorno real y en definitiva es lo que quiere mejorar el cliente que desea el software. La otra dimensión es la que analiza el proceso interno de desarrollar el software. Una vez que se han determinado todos los requisitos del software, el equipo de desarrollo debe lidiar con diversos lenguajes de programación, metodologías, frameworks, paradigmas, artefactos, entre otras muchas tecnologías presentes en un proceso de desarrollo de software. A esta complejidad Brooks la denominó como complejidad *arbitraria*. Esta puede reducirse si se cuenta con las tecnologías y métodos adecuados para enfrentar el desarrollo de un software. Precisamente, en este trabajo se hace un análisis del estado de las propuestas encaminadas a reducir la complejidad arbitraria.

El trabajo está organizado como sigue. En la siguiente sección se desarrolla una valoración de varias propuestas encaminadas a reducir la complejidad arbitraria en Cuba. A continuación, se detalla sobre método de desarrollo de software denominado Arquitectura guiada por modelos (MDA, por sus siglas en inglés). Luego, se presenta una propuesta basada en MDA para el desarrollo de aplicaciones informáticas en Cuba. Finalmente, se presentan las secciones Trabajo futuro y Conclusiones.

Materiales y métodos

Los métodos científicos de mayor relevancia utilizados en este estudio fueron: hipotético-deductivo, histórico-lógico, analítico-sintético, el método sistémico y grupo focal. El método hipotético-deductivo ayudó a elaborar la hipótesis de investigación y proponer líneas de trabajo a partir de resultados de la búsqueda realizada. El método sistémico permitió la concepción integral de una propuesta que tiene en cuenta los modelos en diferentes niveles de abstracción en el desarrollo de software y que todos forman parte de un todo (un sistema) que funcione adecuadamente. El método histórico lógico se utilizó para estudiar la evolución que han tenido los conceptos más importantes que se tratan en esta investigación como es el caso del desarrollo de software, la complejidad arbitraria, los modelos a cada nivel de abstracción, la automatización en el proceso de desarrollo, entre otros importantes conceptos.

El método analítico-sintético se utilizó para realizar un análisis de las principales tendencias del enfoque de Arquitecturas dirigidas por modelos (MDA), su impacto en la reducción de la complejidad arbitraria y finalmente sintetizar los elementos de mayor relevancia para la investigación.

El método de grupo focal se empleó como alternativa para obtener detalles del trabajo en proyectos reales de desarrollo de software. Se contó con especialistas bien calificados que trabajan en distintos tipos de proyectos y con variedad de roles lo que permitió aumentar el conocimiento sobre el tratamiento a la complejidad arbitraria, MDA y cómo son vistas desde un ambiente industrial.

Para obtener los resultados deseados se realizó una búsqueda detallada utilizando el motor de búsqueda *Google Scholar* que es reconocido como líder en la búsqueda de materiales científicos. Con este buscador se obtienen materiales de otras importantes fuentes de datos, entre las que se encuentran ACM, IEEE y Springer que ciertamente son reconocidos como referencias obligadas en las investigaciones de ingeniería del software.

Desarrollo

Análisis de la complejidad arbitraria en Cuba

En esta sección se documenta el análisis desarrollado a partir de una búsqueda realizada sobre la complejidad arbitraria en el desarrollo de software. Es preciso aclarar que el término complejidad arbitraria no ha sido ampliamente generalizado en los trabajos referentes al desarrollo de software. Por ello, en la búsqueda se tuvo en cuenta el significado de complejidad arbitraría más que el término propiamente, es así que se aceptan en la búsqueda otros términos más usados como eficiencia, eficacia, efectividad, agilidad, entre otros términos que sirven para caracterizar el proceso de desarrollo de software. Se han encontrado trabajos que documentan el impacto del software en: Agricultura, Educación, Medicina, Medio Ambiente e Ingeniería Mecánica, entre otros. Sin embargo, en una búsqueda realizada en *Google Scholar*, con las palabras claves 'software + cuba' a partir de 2008, son muy pocos los trabajos que reportan alguna propuesta específica para mejorar el proceso de desarrollo de software. Se analizaron 1000 trabajos resultantes de la búsqueda. De estos trabajos, sólo 15 abordan alguna solución dirigida a mejorar el proceso de desarrollo de software. El resto documenta el impacto que ha tenido algún software en cierta área de la sociedad cubana, destacándose el impacto en la Salud, Educación y Agricultura.

La mayoría de las propuestas encontradas para mejorar el proceso de desarrollo de software se centran en mejorarlo por la vía de perfeccionar la organización de los proyectos. Para ello se proponen metodologías, procedimientos, guías, estándares, entre otras soluciones. Paradójicamente, sólo 8 propuestas reconocen la necesidad de automatizar, al menos parcialmente, el proceso de desarrollo de software. Estos datos podrían llevar a corroborar la frase popular "*en casa del herrero cuchillo de palo*". Pues los desarrolladores de software dedican una buena parte de su tiempo y

esfuerzo a demostrar que los sistemas de software pueden mejorar significativamente cualquier entorno y, sin embargo, no reconocen que el software puede facilitar el propio proceso de desarrollo de software.

Como puede apreciarse en (Ampuero, *et al.* 2010); Revista Ingeniería Industrial; Ampuero, M. A. *et al.* 2009; André *et al.* 2011; Macías and Aguilera, 2011; Varona, *et al.*, 2011; Wilford, and López, 2010), se abordan áreas específicas del proceso de desarrollo de software, pero estas sólo se centran en mejorar algunas actividades del proceso a través del uso de metodologías o procedimientos. Dentro de las áreas que más se abordan están las referidas a la gestión del capital humano dentro de la gestión de un proyecto de desarrollo de software.

Fernández Luna (*et al.*, 2010), reconoce la automatización como una alternativa necesaria para mejorar el desarrollo de software. En ese trabajo se propone un plugin para NetBeans que permite mejorar la colaboración en tiempo real de equipos de desarrolladores que trabajan remotamente. Wilford Rivera (Wilford, 2010), demuestra el impacto que puede tener el uso de la minería de datos en la selección del personal para desarrollar un proyecto de software. También Delgado Dapena (Delgado, 2010) valora el impacto del razonamiento basado en casos para la verificación de la calidad de los sistemas informáticos. Otras propuestas son (Martínez, *et al.*, 2009; Tosca and Fernández, 2009; Wolf, 2011) , donde se aborda alguna temática que de alguna manera puede mejorar el proceso de desarrollo de software y se automatiza alguna actividad dentro del ciclo de desarrollo.

En los trabajos mencionados, se propone la automatización como vía para mejorar el proceso de desarrollo de software, pero no se aborda la obtención de algunos artefactos de forma automatizada. En (Jordán and Vázquez, 2010) se propone obtener el artefacto *Casos de Prueba* a partir de *Casos de Uso*, pero manualmente. Precisamente, la obtención de algunos artefactos a partir de otros resulta uno de los principales retos de la comunidad de investigadores del software. En ese sentido, el paradigma de Desarrollo Dirigido por Modelos (MDD, por sus siglas en inglés) reconoce la importancia de la automatización en el proceso de desarrollo de software y se centra en lograr un producto de software a partir de la transición de modelos.

MDA(OMG, 2003) es una de las propuestas concretas dentro de MDD que mayor aceptación ha tenido dentro de la industria de desarrollo de software. Muestra de ello es la abundante bibliografía sobre la temática, donde se documenta el impacto que ha tenido este enfoque en el proceso de desarrollo. Algunos de estos reportes se encuentran en: (Almeida *et al.*, 2004; Andromda, 2011; Atkinson, 2003; Bocanegra, *et al.*, 2008; Cuesta, *et al.*, 2009; García, *et al.*, 2010; Grangel, *et al.*, 2006; Haitao and Wei, 2010; Kardoš and Drozdová, 2010; Martin and Loos, 2008; McNeile, 2003; Phalp, *et al.* 2007; Sánchez, *et al.*, 2008; Vogel and Mantell, 2006).

Resulta preocupante que a pesar de que MDA es una propuesta que está siendo muy analizada en la comunidad de desarrollo de software desde 2003, se reporten muy pocos trabajos en Cuba sobre esta temática. En la búsqueda realizada sólo en (Martínez,. *et al.*, 2011) se analiza el impacto de este nuevo enfoque en el contexto cubano. No obstante para tratar de obtener un resultado más específico, se realizó una nueva búsqueda, también en *Google Scholar*, donde se especificó como palabras claves *software+cuba+* MDA.

Esta nueva búsqueda arrojó trescientos veinte nueve resultados. De ellos, solamente ocho abordan el enfoque MDA para el desarrollo de software. En la búsqueda aparecieron resultados que no están relacionados con el desarrollo de software, lo que está determinado porque las siglas MDA también son utilizadas en medicina u otra área de investigación.

En (Espinosa, 2012) se ofrecen algunas buenas prácticas para complementar un entorno de desarrollo que sigue el paradigma MDA. En este trabajo no se ofrece una visión completa del proceso de desarrollo de software siguiendo MDA, tampoco se detallan los modelos que se van generando durante todo el ciclo de desarrollo. No obstante, el simple reconocimiento de la factibilidad del paradigma y su complementación con las prácticas que se ofrecen es un paso importante en la extensión de MDA para el desarrollo de software.

En (Martínez, *et al,*. 2011) se realiza un importante estudio sobre las evidencias del impacto que ha tenido MDD y MDA en la industria del software. Este estudio corrobora que MDA es una de las temáticas de mayor interés para la comunidad de investigadores del software. Asimismo, este trabajo muestra el efecto positivo que tiene este paradigma para aumentar la mantenibilidad y usabilidad del software, así como la eficiencia y productividad durante el proceso de desarrollo. Pese a que no se ofrece una solución concreta siguiendo el paradigma MDA, este trabajo constituye un intento relevante por despertar el interés de la comunidad de investigación por el MDD y en particular MDA.

En (Martínez, *et al.*, 2012) se documenta un experimento llevado a cabo por los mismos autores de (Martínez, *et al.*, 2011), donde se realiza una comparación de MDD con los métodos tradicionales de desarrollo de software. Se corroboró que el desarrollo guiado por modelos es considerado más útil que los métodos tradicionales. También los desarrolladores jóvenes se sienten a gusto trabajando con modelos si se asegura un ambiente de desarrollo dirigido por modelos. Como conclusión principal del trabajo se arriba a que el desarrollo dirigido por modelos posee un gran potencial para su adopción en el desarrollo de software de forma generalizada, pero es necesario continuar las investigaciones y las propuestas con el fin de que madure y logre un impacto en el desarrollo de software.

Tras analizar algunas evidencias sobre la necesidad de propuestas para reducir la complejidad arbitraria en el desarrollo de software y demostrar que MDA resulta un paradigma con gran potencial para lograr este fin. Se realiza

en este trabajo una propuesta basada en MDA para el desarrollo de aplicaciones informáticas. Antes, se detalla un poco más el paradigma MDA.

Arquitectura dirigida por modelos (MDA)

MDA es una propuesta promovida por el Object Management Group (OMG, por sus siglas en inglés). En (OMG 2003) se describen las ideas fundamentales de este paradigma, dentro de las que se destacan que es una propuesta para el desarrollo de sistemas y que es dirigido por modelos porque provee los recursos para que los modelos dirijan el curso del entendimiento, diseño, construcción, despliegue, operación, mantenimiento y modificación de los sistemas.

MDA pretende obtener aplicaciones con alta flexibilidad en la implementación, integración, mantenimiento y prueba. Los tres principales objetivos de MDA son: portabilidad, interoperabilidad y reusabilidad. Propone tres puntos de vistas para un sistema: punto de vista independiente de la computación, punto de vista independiente de la plataforma y punto de vista específico de la plataforma.

Un Modelo Independiente de la Computación (CIM): es una vista de un sistema independiente de la computación. No muestra detalles del sistema y está encaminada a reducir la brecha entre los especialistas funcionales y desarrolladores de software.

Un Modelo Independiente de la Plataforma (PIM): es una vista del sistema independiente de la plataforma. Permite usar diferentes plataformas para implementar un sistema.

Un Modelo Específico de la Plataforma (PSM): es una vista de sistema con especificaciones de la plataforma. Un PSM combina especificaciones en los modelos independientes de la plataforma con detalles de cómo el sistema usa ciertos elementos de una plataforma.

La transformación de los modelos es otro elemento clave en MDA. Esta se refiere al proceso de convertir o transformar un modelo a otro del mismo sistema y pudiendo el modelo resultante estar a diferente nivel de abstracción que el modelo de origen. Uno de los principales esfuerzos de la comunidad de investigadores es lograr que la transformación se realice de la forma más automatizada garantizando la calidad de los modelos.

Análisis de las causas de la poca atención a la complejidad arbitraria y MDA

Como se mencionó, y teniendo en cuenta el resultado de la búsqueda realizada, resulta un tanto insólito el poco interés prestado a la reducción de la complejidad arbitraria y el empleo de MDA en Cuba. En la propia literatura referente a MDA se pueden encontrar algunas de las causas de este fenómeno. Por ejemplo Bran Selic en (SELIC

2008) le atribuye el impacto por debajo de las expectativas creadas por MDA a tres obstáculos fundamentales : *técnicos, culturales y económicos.*

Para examinar este tema en la universidad se aplicó una técnica que permitió conocer el pensamiento de un grupo de especialistas con experiencia en el área de la ingeniería y desarrollo del software, sobre las causas que inciden en el poco interés sobre MDA en Cuba. La técnica aplicada fue de *Grupo focal*. En el grupo existían profesionales vinculados con el área académica, la producción y la investigación. Para conformar el grupo se tuvo en cuenta que poseyeran diferentes categorías científicas, quedando conformado de la siguiente manera: un doctor en ciencias, tres másteres y tres estudiantes de maestría. En el análisis del grupo emergieron algunas razones muy interesantes, por ejemplo, se manifestó que los desarrolladores de software generalmente se enfocan en resolver sus problemas y muchas veces llegan a soluciones de interés científico pero no les interesa publicarlos. Por eso, una conclusión importante del debate fue que la información recogida sólo a través de artículos científicos, aunque ofrece una idea del estado de la industria de software, no es suficiente para obtener una caracterización cercana a la realidad. En análisis se evidenció la necesidad de aplicar instrumentos para la captura de información en los principales centros de desarrollo del país, mediante un proceso organizado de Gestión del conocimiento que pueda ser utilizado en la toma de decisiones sobre el desarrollo de software.

Otra de las ideas emergidas del debate fue que los desarrolladores de software no asumen con mayor interés MDA porque no perciben un posible incremento de su productividad. Un modelo de datos, por ejemplo, es muy difícil que algún desarrollador se aventure a realizarlo sin antes haber realizado el modelo *Entidad-Relación*. Sin embargo, no sucede lo mismo con otros artefactos, por ejemplo, un diagrama de clases donde los desarrolladores asumen, con cierta lógica, que pueden codificar directamente sin partir de un diagrama de clases y hacerlo bien.

Coincidiendo con Selic se identificó que la dinámica de los proyectos de desarrollo de software, donde es necesario obtener resultados en muy poco tiempo y a menudo los desarrolladores no lo poseen para hacer estudios sobre las tecnologías con mayores posibilidades, se limitan a utilizar las que ya dominan.

Los autores de este articulo consideran que es imprescindible seguir trabajando para demostrar, sobre la bases de las teorías que lo sustentan y de ejemplos de aplicación práctica, que MDA es aplicable con buenas perspectivas en la industria de software. Sobresalen dos líneas fundamentales, la primera es realizar investigaciones que le brinden un respaldo teórico importante a MDA. La otra línea consiste en divulgar las experiencias positivas de la aplicación de MDA en la industria de desarrollo y en correspondencia con ello es necesario realizar una correcta política de gestión del conocimiento en los centros de desarrollo de software en el país. En consecuencia con eso, a continuación se

ofrece un método basado en MDA que puede ser de gran utilidad, en otros trabajos se documentará el impacto de su aplicación.

Método para el desarrollo de aplicaciones basado en MDA

En esta sección se describe una propuesta concreta para el desarrollo de software basado en MDA. Para ello, se describen los modelos en los diferentes niveles de abstracción, comenzando por los modelos independientes de la computación. El método que se propone está dirigido principalmente al desarrollo de sistemas de gestión empresarial. No obstante, puede ser aplicado al desarrollo de sistemas en diferentes dominios de aplicación.

Modelo independiente de la computación (CIM): El objetivo fundamental de los modelos que componen este nivel es lograr una caracterización lo más real posible del entorno en donde se va desplegar el software. Con este fin, la representación de los procesos de negocio que se desarrollan en el entorno de aplicación es recomendable.

Según (OMG, 2008) un proceso es "*un tipo de comportamiento interactivo que describe actividad(es) específica(s) a ser desarrolladas, las interacciones a ser realizadas durante su ejecución bajo la autoridad de un Rol Procesador (o roles ejecutantes delegados). El proceso posee el conjunto de actividades a ser desarrolladas así como las condiciones por las cuales las actividades se desarrollarán y por qué rol. El proceso también posee las Partes Interactivas que definen el flujo de información y otros recursos entre actividades, Rol ejecutor y Rol de interacción*".

Existen varios lenguajes o métodos para modelar procesos de negocio, entre los que se encuentran: Métodos para el modelado funcional (IDEF0), Lenguaje Unificado para la construcción de Modelos(Lazar, *et al.*, 2010), Notación para el Modelado de Procesos de Negocio (BPMN), entre otros. En los últimos tiempos BPMN ha emergido como la propuesta con mayor aceptación, porque define una notación fácilmente entendible, tanto por todos los involucrados como por el equipo de desarrollo (OMG, 2009).

Un componente esencial en BPMN es el diagrama de procesos de negocio (BPD), el cual es usado para crear modelos gráficos de procesos de negocio. Los elementos fundamentales de un BPD descritos en (White, 2004) son: Objetos de flujo, Objetos de conexión, Líneas de vida y Artefactos. Por las notables ventajas que ofrece esta notación se selecciona para representar el modelo independiente de la computación.

Modelo independiente de la plataforma (PIM): La arquitectura de sistema provee una vista de alto nivel de abstracción, que muestra una división del sistema en componentes, las interacciones que ocurren entre ellos se representan en un modelo de componentes. Para obtener una vista del sistema independiente de la plataforma se propone la vista de arquitectura de sistema, su propósito fundamental es mostrar cómo los componentes del sistema se

interrelacionan para dar cumplimiento a las funcionalidades identificadas durante el modelado de los procesos de negocio. El modelo de componentes propuesto está constituido por tres elementos principales:

- Componente: Es la unidad fundamental de la vista lógica del sistema, que representa una abstracción de una parte del sistema donde se implementan algunas funcionalidades.
- Servicios: Es el punto de enlace que permite la colaboración entre los componentes para satisfacer las funcionalidades requeridas en el sistema.
- Funcionalidad: Constituyen los requisitos que debe implementar cada componente.

Es necesario aclarar que este modelo está en correspondencia con el estilo arquitectónico orientado a componentes. No obstante, en un modelo de componentes no se podrán recoger todos los aspectos necesarios para obtener modelos específicos de una plataforma. Una de las vistas necesarias en este nivel de abstracción es la vista de datos.

De esta manera el modelo de datos será el segundo modelo a nivel PIM que se propone. Precisamente la vista de datos es una de las sugeridas por la mayoría de las propuestas para la descripción de la arquitectura de software. Específicamente se desarrollará un modelo Entidad-Relación. En este modelo se identifican las tablas que guardan los datos que se necesita persistir y las relaciones entre ellas para que el sistema funcione correctamente y realice las funciones que se esperan de él. Se debe especificar que aunque son dos modelos diferentes en el mismo nivel de abstracción, están relacionados entre sí. Los componentes están asociados a diferentes tablas en el modelo de datos. Además, esta relación será importante para la representación de modelos con menor nivel de abstracción.

Modelo específico de la plataforma (PSM): Al inicio de la sección se explicó que esta propuesta estaba encaminada al desarrollo de aplicaciones empresariales. Por ello, es conveniente que para el nivel PSM se cuente con una plataforma de desarrollo que se especialice en el desarrollo de este tipo de sistemas. Aunque existen varias plataformas que podrían ajustarse a esa característica, para esta propuesta se seleccionó el framework SAUXE. En (Camejo and Gálvez, 2010) se detallan las características de este framework que determinaron su selección.

El nivel PSM contará con el mismo modelo de componentes que a nivel PIM, pero se le incorporan detalles propios de SAUXE. Por ejemplo, para la integración de componentes en SAUXE se utiliza el protocolo *Inversión of control (IOC)*. Para ello, se define una clase *ioc* la cual se encarga de abstraer a los programadores del proceso de integración entre componentes. Por lo tanto, en el modelo de componentes para representar la integración de componentes se especifica que es a través de la clase *ioc.*

Además, es necesario también obtener un modelo de datos específico de una tecnología. En este caso, SAUXE está implementado para utilizar el gestor postgres. Esto significa que al modelo de datos independiente de la plataforma es

necesario incorporarle los detalles propios de postgres. De esta forma, el nivel PSM también está integrado por dos modelos: el modelo de componentes específico de SAUXE y el modelo de datos específico de postgres.

En SAUXE se define una estructura para los componentes. Cada componente estará formado por: *modelo, vista, controlador y servicios*. En la *vista* se implementan las interfaces de usuario, en caso que el componente posea. El *controlador* se encargará de ofrecerle los datos a la *vista* y servir de mediador con el *modelo*. En el *modelo* se realizarán las operaciones propias del componente y se implementa el acceso a los datos. A partir de las relaciones especificadas en el nivel PIM entre los componentes y las tablas se podrá generar automáticamente las clases del modelo que se encargan del acceso a los datos directamente con el gestor. Por último, en la capa *servicios* se definirán todos los servicios que consume y provee el componente. Esta información también se obtiene a partir del modelo de componentes independiente de la plataforma. La figura 1 muestra una vista general de la propuesta, donde se muestran los modelos propuestos a cada nivel.

Con la definición de los modelos y las transformaciones entre ellos preservando las interfaces permite que los modelos puedan evolucionar independientemente, lo que agiliza el proceso de desarrollo ante ciertos cambios. Estos cambios pueden aparecer en dos variantes: evolución de la tecnología o modificación en la lógica de negocio. Si se sigue esta propuesta ante una evolución de la tecnología el cambio es mínimo, porque sólo se actualizan ciertos elementos de los modelos a partir del nivel PSM, quedando al margen de estos cambios los modelos de mayor nivel de abstracción. Por el contrario, si varía la lógica de negocio entonces se actualizarán todos los modelos, pero como se deben definir reglas para la transición entre modelos entonces la actualización se podrá realizar de forma automática, sólo es necesario preservar la consistencia de las transformaciones.

La adopción de esta propuesta permitiría organizar mejor el proceso de desarrollo. Con la determinación de los modelos que se desarrollan en cada nivel de abstracción se asegura la homogenización del proceso de desarrollo. Además, el establecimiento explícitamente de la manera de obtener cada elemento dentro de un modelo evitaría errores provocados por la subjetividad de los involucrados. La productividad y eficiencia del proceso de desarrollo son dos de las variables que se pueden mejorar si se aplica esta propuesta, además se mejoraría la calidad, mantenibilidad y reutilización del sistema desarrollado. Pese a que la propuesta está dirigida al desarrollo de sistemas de gestión empresarial puede ser adaptada para el desarrollo de otro tipo de sistemas, por ejemplo el desarrollo de sistemas de salud pueden beneficiarse con la adopción de una propuesta como esta. Evidencias empíricas (Martínez, *et al., 2011*) demuestran que con la ayuda de métodos de desarrollo dirigido por modelos la mantenibilidad y productividad en el proceso de desarrollo se ven favorecidas así como la calidad de los sistemas desarrollados.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 156-171
http://rcci.uci.cu

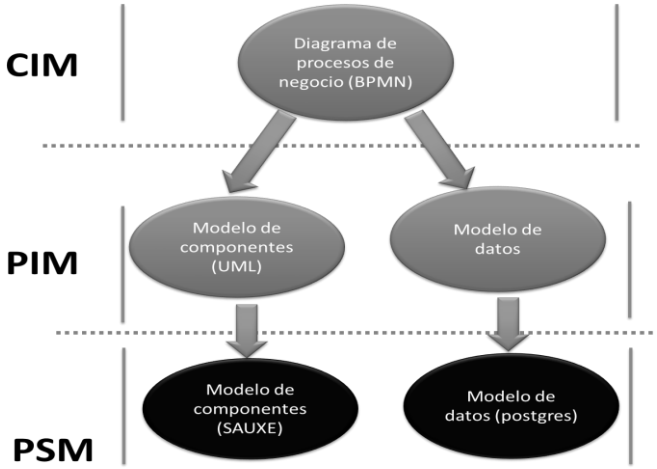

Figura 1. Vista general de la propuesta basada en MDA. Fuente: Elaboración propia.

Discusión

Con la revisión realizada y complementado con el grupo focal ejecutado se comprobó que no existe una adecuada aplicación de los métodos más novedosos para el desarrollo de software en Cuba. Ni la industria de software ni las investigaciones relacionadas con el desarrollo de software en Cuba se han interesado por MDA pese a que acapara gran aceptación por la comunidad de investigadores y desarrolladores de software en el mundo.

Teniendo en cuenta las evidencias descritas en la literatura sobre el impacto positivo de MDA en la productividad del proceso de desarrollo y en la mejora de la calidad de los productos en este trabajo se han definido los modelos que componen un método de desarrollo basado en MDA. Una de las líneas en las que se debe profundizar es en la transformación de los modelos. Además, se deben hacer esfuerzos en lograr que las transformaciones se realicen de forma automática. Para lograr la transformación de CIM a PIM será necesario realizar un proceso investigativo profundo, ya que precisamente es una de las líneas de investigación abiertas en la temática de MDA.

Como conclusión de la revisión de la literatura especializada el otro elemento clave en aras de perfeccionar el método, es la validación de cada uno de los modelos. Será necesario el uso de lenguajes formales para la representación de modelos. Debe comprobarse la consistencia de los modelos, tanto los que están al mismo nivel de abstracción como en diferentes niveles. Las ontologías resultan una variante con muchas posibilidades. En los últimos años se ha incrementado su uso en el desarrollo de software, sobre todo en las tecnologías vinculadas con la web semántica donde se ha demostrado su capacidad en la validación de modelos y el chequeo de la consistencia de forma automática (W3C, 2006).

Conclusiones

En este trabajo se realizó una valoración sobre la necesidad de realizar investigaciones y lograr propuestas concretas encaminadas a reducir la complejidad arbitraria en el desarrollo de sistemas de software. A partir de una búsqueda realizada y del análisis de los resultados se pudo comprobar que en Cuba no se da prioridad a investigaciones de este tipo. El examen de las potencialidades del enfoque MDA para reducir la complejidad arbitraria, demostró las posibilidades de esta propuesta para mejorar el proceso de desarrollo de software. Sin embargo, en Cuba se reportan muy pocos trabajos que documenten el impacto que tiene este paradigma en el desarrollo de software. Considerando estos elementos se presentó un método para el desarrollo de software basado en MDA y dirigido al desarrollo de sistemas de gestión empresarial. Esta propuesta constituye un paso inicial para lograr un método más completo que tenga un impacto significativo en la productividad y eficiencia durante el proceso de desarrollo, así como en la calidad del producto final. Para completar este método se sigue trabajando en las líneas descritas en la sección correspondiente a trabajo futuro.

Referencias

- ALMEIDA, J. P.; DIJKMAN, R., *et al. On the Notion of Abstract Platform in MDA Development*. Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conf (EDOC 2004), 2004 IEEE, 2004.
- AMPUERO, M. A.; BALDOQUÍN DE LA PEÑA, M. G., *et al.* Propuesta de roles invariantes y competencias para enfrentar proyectos de software *Revista Ingeniería Industrial*, 2010, 30(2).
- AMPUERO, M. A.; GULNARA BALDOQUÍN DE LA PEÑA, M., *et al.* Gestión del conocimiento para la elaboración de un modelo formal de asignación de personal a equipos de proyectos de software *Revista de Ingeniería Industrial.* 2009, 30(2).
- ANDRÉ, M.; BALDOQUÍN, M. G., *et al.* Formal model for assigning human resources to teams in software projects *Information and Software Technology*, 2011, 53(3): p. 259-275.
- ANDROMDA. *AndroMDA*, 2011. [2011]. Disponible en: [http://www.andromda.org].
- ATKINSON, C. The Role of Metamodeling in MDA, 2003.
- BOCANEGRA, J.; PEÑA, J., *et al. Una Aproximación MDA para Modelar Transacciones de Negocio a Nivel CIM*. Jornadas de Ingeniería del Software y Bases de Datos, España, 2008. p. 1988–3455
- BROOKS, F. P. J. *The Mythical Man-Month*. Anniversary Edition. University of North Carolina at Chapel Hill, ADDISON-WESLEY, 1995. 0201835959.

- CAMEJO, R. R. B. and GALVEZ, A. T. *Herramienta para la gestión y el análisis de trazas en el sistema de gestión integral CEDRUX*. *GESTEC* Habana, Cuba, 2010.
- CUESTA, M. A.; LÓPEZ, T. M., *et al.* Comparativo de herramientas MDA (AndroMDA, ArcStyler, Op timalJ) *Vector*, 2009, Volumen 4, p. 50 - 58.
- DELGADO DAPENA, M. Calidad de los proyectos de software: revisiones utilizando razonamiento basado en casos *Revista Ingeniería Industrial*, 2010, 24(2).
- ESPINOSA, M. M. Buenas prácticas sobre gestión de configuración en proyectos con metodología MDA *Revista Cubana de Ingeniería*, 2012, 3(2): p. 43-49.
- FERNÁNDEZ-LUNA, J. M.; HUETE, J. F., *et al. COSME: A NetBeans IDE plugin as a team-centric alternative for search driven software development*. *CIS '10* Florida, USA, 2010.
- GARCÍA-DÍAZ, V.; FERNÁNDEZ-FERNÁNDEZ, H., *et al.* Talisman MDE: Mixing MDE Principles *Journal of Systems and Software*, 2010, 83(7): 1179-1191.
- GRANGEL, R.; BOUREY, J.-P., *et al.* Solving Problems in the Parameterisation of ERPs Using a Model-Driven Approach, 2006.
- HAITAO, W. and WEI, C. *Research on Modeling and Model Converting Approaches Based on MDA*. *Second WRI World Congress on Software Engineering*, IEEE, 2010.
- JORDÁN ENRÍQUEZ, O. and VÁZQUEZ RUIZ, O. Generación de Casos de Prueba a partir de casos de Uso en las pruebas de software *Revista Ingeniería Industrial*, 2010, 27(1).
- KARDOŠ, M. and DROZDOVÁ, M. Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA) *JIOS*, 2010, 34, NO. 1: p. 89-99.
- LAZAR, I.; S. MOTOGNA, *et al.* Behaviour-Driven Development of Foundational UML Components. Electronic Notes in Theoretical Computer Science *Theoretical Computer Science*, 2010: p. 91-105.
- MACÍAS GELABERT, C. R. and AGUILERA MARTÍNEZ, A. F. Modelación de la dependencia y estructura del conocimiento en procesos de trabajo: Una aplicación en la industria cubana del software *Revista Facultad de Ingeniería Universidad de Antioquia*, 2011: p. 219-226.
- MARTIN, A. and LOOS, P. Software support for the Computation Independent Modelling in the MDA context, 2008.
- MARTÍNEZ DEL BUSTO, M. E.; IBARGOLLÁN PÉREZ, N., *et al.* Edición de reglas de negocio sobre una ontología que permita su implementación independiente de las bases de datos *Revista Cubana de ciencias informáticas*, 2009, 3(1-2).

- MARTÍNEZ, Y.; CACHERO, C., *et al.* Evidencia empírica sobre mejoras en productividad y calidad en enfoques MDD: un mapeo sistemático *REICIS Revista Española de Innovación, Calidad e Ingeniería del Software*, 2011, 7(2): p. 6-27.
- MARTÍNEZ, Y.; CACHERO, C., *et al.* MDD vs. Traditional Software Development: a practitioners subjective perspective *Information and Software Technology*, 2012, (0).
- MCNEILE, A. MDA: The Vision with the Hole? 2003.
- NAUR, P. A. R., B, Ed. *Software Engineering: Report of a conference sponsored by the NATO Science Committee*Garmisch, Germany, Scientific Affairs Division, NATO, 1968.
- OMG. *Business Process Definition MetaModel Volume II: Process Definitions*, 2008.
- OMG. *Business Process Model and Notation (BPMN). Versión 1.2*, OMG, 2009.
- OMG. *MDA Guide Version 1.0.1*. 2003.
- PHALP, K.; JEARY, S., *et al.* Supporting Stakeholders in the MDA Process, 2007.
- SÁNCHEZ VIDALES, M. Á.; FERMOSO GARCÍA, A., *et al.* Una recomendación basada en MDA, BPM y SOA para el desarrollo de software a partir de procesos del negocio en un contexto de Negocio Bajo Demanda, 2008.
- SELIC, B. Manifestaciones sobre MDA *Novática*, 2008, (192).
- TOSCA, F. G. and FERNÁNDEZ, R. M. Reingeniería de Software ¿Un camino o el camino? *Revista Internacional la Nueva Gestión Organizacional*, 2009, 4(8): p. 13-25.
- VARONA, D.; CAPRETZ, L. F., *et al.* Personality Types of Cuban software developers *Global Journal of Engineering Education*, 2011, 13(2): p. 77-81.
- VOGEL, R. and MANTELL, K. *MDA adoption for a SME: evolution, not revolution – Phase II. ECMDA Workshop*, 2006.
- W3C, E. P. T., JEFF Z. PAN, DANIEL OBERLE, EVAN WALLACE, MICHAEL USCHOLD, BOEING, ELISA KENDALL. *Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering*, 2006. [2011]. Disponible en: [http://www.w3.org/2001/sw/BestPractices/SE/ODA/
- WHITE, S. A. I. C. Introduction to BPMN, 2004.
- WILFORD RIVERA, I. Minería de Datos: Herramienta de Apoyo en la Selección de Equipos de Proyectos Informáticos *Ingeniería Industrial*, 2010 27(2-3).
- WILFORD RIVERA, I. and LÓPEZ TRUJILLO, Y. Uso Efectivo de los Recursos Humanos Implicados en el Proceso De Desarrollo De Software *Revista Ingeniería Industrial*, 2010, 27(2-3): p. 4.

- WOLF, G. Monitoreo de PostgreSQL con Munin *Revista Cubana de Ciencias Informáticas*, 2011, 5(1): p. 1-8.