Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Desarrollo de Aplicaciones Informáticas
Recibido: 14/11/2013 | Aceptado: 7/04/2014

Framework basado en MDA y ontologías para la representación y validación de modelos de componentes

Framework based on MDA and ontology for the representation and validation of components model

Nemury Silega-Martínez [1]*, Danaysa Macías-Hernández [2], Yusnier Matos[1], Juan Pedro Febles[3]

[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. C.P.: 19370 Cuba
[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. C.P.: 19370 Cuba
[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. C.P.: 19370 Cuba
* Autor para correspondencia: nsilega@uci.cu; ymarias, dmacias, febles@uci.cu

Resumen

Arquitectura Dirigida por Modelos (MDA) es una de las propuestas más prominentes en el área del desarrollo de software, cuenta con aceptación tanto en la comunidad de investigadores como en la industria de desarrollo. Por otra parte, en los últimos años se ha demostrado las potencialidades de las ontologías para la representación de un dominio determinado, muestra de ello son los resultados en la web semántica. En este trabajo se presenta una propuesta basada en el paradigma MDA y que se complementa con una ontología para representar y validar modelos de componentes. Este modelo de componentes se restringe al desarrollo de sistemas de gestión empresarial, por eso incluye conceptos propios de ese dominio. El uso del *framework* reducirá la cantidad de errores cometidos durante el desarrollo de la arquitectura de sistemas así como aumentará la estandarización y productividad en esta fase.

Palabras clave: Arquitectura dirigida por modelos, modelo de componentes, ontología, sistemas de gestión empresarial.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

Abstract

Model Driven Architecture is one of the most prominent proposals in the area of software development, accepted by both the research community and software development industry. Moreover, in recent years have shown the potential of ontologies for representing a particular domain, example of this are the results in the semantic web. In this paper we present a proposal based on Model Driven Architecture paradigm and is complemented with ontology to represent and validate component models. This component model is restricted to the development of business management systems, so it includes concepts from that domain. The use of the framework will reduce the number of errors made during the development of the system architecture, will increase standardization and productivity at this stage.

Keywords: *Components model, enterprise management systems, model driven architecture, ontology.*

Introducción

Los sistemas de gestión empresarial se encargan de informatizar los procesos que se realizan en las empresas. Un tipo particular de sistemas de gestión empresarial son los sistemas de Planificación de Recursos Empresariales (ERP por sus siglas en inglés). Los ERP gestionan de manera integrada y eficiente los recursos de una o varias empresas, con el objetivo fundamental de maximizar las ganancias de las mismas. Estos sistemas están caracterizados por ser: integrados, modulares y configurables (Carreón, 2008). Estas características determinan que el desarrollo de un ERP sea una actividad compleja y que finalmente, según (Benvenuto, 2006) su solución se caracterice por: elevado número de funcionalidades, adaptabilidad, modularidad, orientación a los procesos de negocio y universalidad.

Las características descritas determinan que durante todo el desarrollo de un sistema de gestión empresarial es necesario estandarizar todos los aspectos del ciclo de vida del desarrollo, en especial, la construcción de la arquitectura de sistema. La arquitectura de sistema provee una vista de alto nivel de abstracción en un contexto de desarrollo orientado a componentes, muestra una división del sistema en componentes y sus interacciones, las que se representan en un modelo de componentes. Su propósito es mostrar cómo los componentes del sistema se interrelacionan para satisfacer las funcionalidades identificadas en el modelado de los procesos de negocio.

La relevancia de la arquitectura de sistema fija la necesidad de contar con los mecanismos necesarios tanto para la representación como para la validación. En este trabajo se presenta una propuesta teórica para la descripción y validación de modelos de componentes que aprovecha las potencialidades del paradigma MDA y la complementa con una ontología. Esta propuesta facilitará la uniformidad de los componentes del sistema y su extensión dentro del contexto MDA contribuiría a estandarizar la lógica de diseño, elevar el nivel de reutilización y aumentar la productividad en el proceso de desarrollo. Por otra parte, el uso de una ontología, como notación formal para

representar el modelo de componentes, contribuirá a la automatización en la actividad de validación y permitirá intercambiar y compartir el modelo de componentes para que pueda ser analizado por herramientas o sistemas inteligentes capaces de leer la ontología.

Materiales y métodos

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. De acuerdo a (OMG, 2003) los tres objetivos principales 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 reduce 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 forma más automatizada garantizando la calidad de los modelos.
En las propuestas descritas en (Bocanegra, Peña, *et al.*, 2008; Singh and Sood, 2010; De Castro, Marcos, *et al*., 2010; Sánchez Vidales, Fermoso García, *et al.*, 2008; Mora, García *et al*., 2008) se demuestra el impacto positivo de la aplicación de aproximaciones MDA en el desarrollo de software. Especialmente en (Martínez, Cachero *et al*., 2011)

se realiza un estudio de otras propuestas donde se recogen evidencias empíricas del efecto de la aplicación de enfoques MDA en el desarrollo de software. De ese estudio se concluye que el uso de las definiciones MDA contribuye a mejorar la productividad y calidad del proceso de desarrollo así como a elevar la calidad de los productos desarrollados. También en (Martínez, Cachero, *et al*., 2012) se realiza un experimento para comparar la aceptación del desarrollo dirigido por modelos con métodos tradicionales, los resultados demuestran el creciente interés en los métodos de desarrollo dirigido por modelos. La consideración de estas propuestas determinó que en este trabajo se adoptara MDA como base de la propuesta.

Estilo arquitectónico para sistemas de gestión empresarial

En (Matos and Silega, 2013) se propone un estilo arquitectónico específico para sistemas de gestión empresarial. Ese estilo arquitectónico se basa en las propuestas de (Garlan, 2003; Taylor, Medvidovic, *et al*., 2009; Kruchten, 2004). El estilo incorpora experiencias que han resultado positivas en el desarrollo del sistema ERP CEDRUX (Lage, Silega et al. 2009) y que son generalizadas para su aplicación en cualquier sistema de gestión empresarial.

Ontologías

En (Noy and McGuinness, 2001) se define ontología como: Una descripción formal explícita de los conceptos (clases) en un dominio de discurso, las propiedades de cada concepto describen sus rasgos, atributos y restricciones.

La utilización de ontologías puede complementar el paradigma MDA, al posibilitar la representación de vocabularios de dominio no ambiguos, el chequeo de la consistencia de los modelos, validación y nuevas capacidades. La aplicación de ontologías junto con MDA ha dado como resultado una nueva propuesta, denominada Arquitectura Dirigida por Ontologías (ODA) (W3C, 2006).

Existen varios lenguajes para especificar ontologías, dentro de los que se destacan: *Ontolingua*, *XML Schema*, *RDF* (del inglés, *Resource Description Framework*), *RDF Schema* (o *RDF-S*) y *OWL (del inglés, Ontology Web Language)* (Xing and Ah-Hwee, 2010). Dentro de todos, se distingue OWL por las facilidades que brinda y dentro de las que sobresale su conjunto de operadores: intersección, unión y negación (Horridge, 2009). Está basado en un modelo lógico que le permite definir los conceptos tal y como son descritos. Además, la posibilidad de utilizar razonadores permite chequear automáticamente la consistencia de los modelos representados. Las ventajas descritas y la posibilidad de contar con la herramienta Protégé que permite crear ontologías de manera sencilla en OWL y utilizar razonadores, han determinado que en esta propuesta se asuma OWL como el lenguaje para la representación de ontologías.

Las propuestas presentadas en (Pahl, Giesecke, *et al*., 2009; Bo and Li-juan, 2009; Chengpu, Rob, *et al*. 2010; Chungoora and Young 2008; Kruchten, 2004) demuestran las potencialidades de las ontologías en la ingeniería del software, en especial para la descripción y validación de las arquitectura de software.

Framework basado en MDA y ontologías para la representación y validación de modelos de componentes

El framework considera los modelos que componen el nivel de los PIM para representar la arquitectura de sistema. Como propone MDA, se comenzará por la descripción del metamodelo definido. En este caso se propone un metamodelo basado en el estilo arquitectónico descrito en (Matos and Silega, 2013) y se enriquece con nuevos elementos. Los elementos del metamodelo son:

- *Componente:* Unidad fundamental de la vista lógica, muestra una abstracción de una parte del sistema que implementan algunas funcionalidades.
- *Servicio:* Es el punto de enlace que permite la colaboración entre los componentes para satisfacer las funcionalidades requeridas en el sistema.
- *Funcionalidad:* Requisitos que debe implementar cada componente, representan la interacción del usuario con el sistema.

A su vez los componentes reciben una clasificación en dependencia de su función dentro del sistema o de las características propias del componente:

- *Componentes de negocio:* Componentes con la responsabilidad de abstraer las principales funcionalidades del negocio de la organización, comúnmente agrupan el deseo funcional del sistema que solicita el cliente.
- *Componentes del dominio:* Componentes con la responsabilidad de abstraer las configuraciones y parametrizaciones estáticas o dinámicas del sistema. Generalmente están asociados a requisitos especiales o restricciones del sistema, por ejemplo un requisito característico en algunos sistemas es que se puedan realizar operaciones en diferentes monedas. Los componentes de este tipo suelen servir de soporte para que los componentes de negocio realicen sus procesos, que en última instancia es lo que más le interesa al cliente.
- *Componentes tecnológicos:* Representan aquellos componentes con la responsabilidad de abstraer características tecnológicas que sirven de base al resto de los componentes del sistema. Tienen la responsabilidad de contribuir directamente con la abstracción de las características no funcionales. En estos componentes se van a agrupar los requisitos que luego deben satisfacer cuando se transformen a componentes de una plataforma específica. Un componente imprescindible dentro de esta categoría, es aquel que tiene la responsabilidad de abstraer la estrategia

de integración entre los componentes del sistema, a este se le denominará componente de integración. En otro nivel de abstracción se deben introducir nuevos componentes de integración, en caso de que se necesite interoperar con componentes en diversas plataformas.

Los componentes se clasifican según su tamaño, dependencia o relevancia dentro del sistema en:

- *Núcleo:* Componentes de los que se consume un número importante de servicios.
- *Complejo:* Implementan un número importante de funcionalidades.
- *Cliente:* Consumen algún servicio que provee otro componente.
- *Proveedor:* Provee servicios a otros componentes.
- *Independiente:* No consume ningún servicio.

La separación de los componentes en dependencia de su función dentro de un sistema ofrece importantes ventajas. Dentro de las más notables se encuentra la posibilidad de organizar objetivamente el orden de desarrollo de los componentes, comenzando por los componentes de dominio. La reutilización también se ve favorecida por esta propuesta, como los componentes de dominio no deben tener dependencias pueden ser reemplazados fácilmente sin causar grandes perjuicios al sistema, sólo se debe respetar los servicios que brindan. Por otra parte, también se facilita y agiliza el diseño, como previamente se conocen los tipos de componentes los diseñadores pueden separar los requisitos del negocio y los requisitos del dominio y luego asociarlos a componentes de negocio y dominio respectivamente. En general con esta clasificación se facilita la gestión de proyectos, aumenta las posibilidades de reutilización y se agiliza el diseño de alto nivel.

En el metamodelo se incluyen los conceptos *Elementos patrimoniales* y *Elementos de dominio* que son propios de los procesos de gestión empresarial (Blanco, 2008). La inclusión de estos conceptos en un modelo de componentes permite relacionar directamente elementos del negocio con elementos del sistema. Además, contribuye a la modularidad del diseño porque cada componente debe implementar sólo las funcionalidades asociadas a un elemento patrimonial o de dominio. De esta forma se facilita la actualización o sustitución de un componente determinado y el impacto en el sistema es mínimo.

La utilización de este metamodelo permite que los modelos de componentes sean descritos a partir de los mismos conceptos. Además, como parte del metamodelo también se establecen pautas de cómo debe ser la relación entre los conceptos de cada modelo. Estos elementos reducen el impacto de la subjetividad de los arquitectos y aumenta la

uniformidad entre los diferentes modelos de componentes aunque sean realizados por personas distintas ya que cuentan con los mismos elementos de modelado.

También se incluyen restricciones que determinan la interacción entre los elementos del metamodelo, como son:

R1. Los componentes de dominio no pueden ser dependientes de otros componentes.

R2. Un componente solo puede estar asociado a un elemento patrimonial o de dominio.

R3. Todas las funcionalidades deben estar asociadas a un componente y sólo un componente.

R4. Todos los componentes deben implementar alguna funcionalidad.

R5. Los servicios deben estar asociados a una funcionalidad y sólo una funcionalidad.

Modelo ontológico: El primero de los modelos que se concibe para describir el PIM es una ontología. Se desarrolló una ontología que contiene los principales elementos del metamodelo, se representan como clases los *Componentes*, *Servicios* y *Funcionalidades* mediante las clases *Component*, *Service* y *Functionality* respectivamente. Se incluyen propiedades que relacionan los conceptos del modelo de componentes, por ejemplo se especifica que un *Servicio* es provisto por algún *Componente* y que un *Componente* implementa ciertas *Funcionalidades*. En la Figura1 se muestran las clases de la ontología.

La ontología contiene propiedades (*object properties*) que permiten relacionar las clases del modelo. Por ejemplo, la propiedad *hasfunctionality* indica la relación entre un componente y una funcionalidad determinada. La propiedad *SupportTo* indica que un componente informatiza un *elemento patrimonial* o de *dominio* determinado. La propiedad *ProvideService* establece que un componente provee un servicio determinado. Cada una de las propiedades tiene su propiedad inversa, en este caso las propiedades *IsFunctionalityOf, IsSuportBy y IsProviededBy* son inversas de *hasfunctionality, SupportTo y ProvideService* respectivamente.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

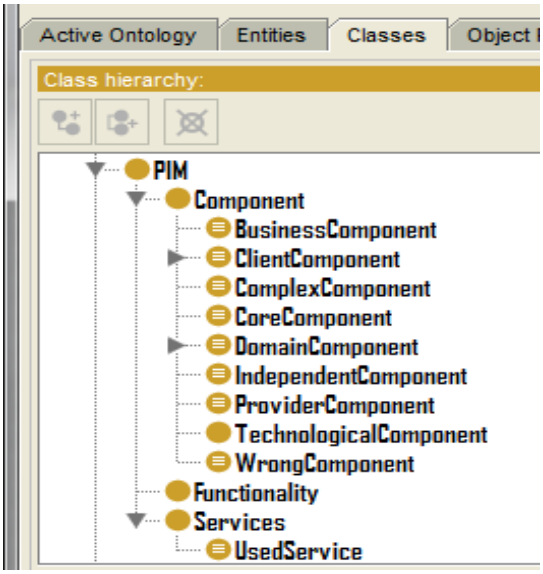

Figura 1. Clases del Modelo de Componentes.

En la ontología se incluyen clases para la clasificación de los componentes, una clase para cada tipo de componente como se muestra en la Figura 2. También se incluye una clase denominada *WrongComponent* que contendrá a los componentes que violan las restricciones definidas en (Matos and Silega, 2013). En la ontología también se incluyen propiedades que permiten verificar el cumplimiento de las restricciones del metamodelo.

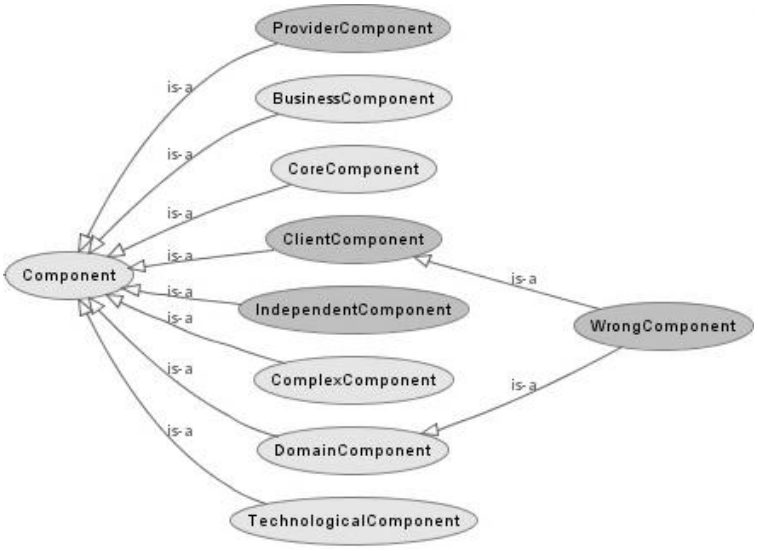

Figura 2. Tipos de Componentes.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

En la ontología se establecieron mecanismos para comprobar las restricciones del metamodelo. Como muestra la Figura 3 se especificó que todo componente debe tener al menos una funcionalidad y que debe informatizar un elemento patrimonial o de dominio. Con esas especificaciones se podrán validar las restricciones R1 y R2.

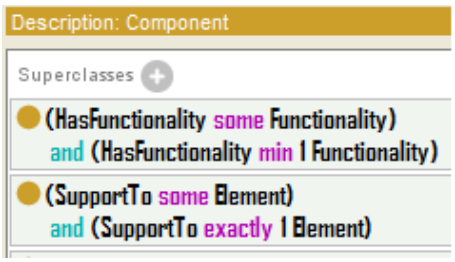

Figura 3. Restricción para la clase *Component.*

Modelo de componentes UML: Tener expresado un modelo de componentes en una ontología ofrece importantes beneficios como se describió anteriormente. Sin embargo, resulta difícil de analizar por los humanos. Por ese motivo, una vez que el modelo de componentes sea validado en la ontología se transforma a otro con mejor representación visual, amigabilidad y que facilite la comprensión por los humanos. En este caso se selecciona el diagrama de componentes UML. Por lo tanto el nivel CIM está compuesto por un modelo de componentes formal expresado mediante una ontología y finalmente un modelo de componentes UML.

Resultados y discusión

Clasificación de componentes

Un componente se va a considerar complejo cuando implemente cierta cantidad de funcionalidades, para ilustrar las posibilidades de clasificación automática en la ontología, en el ejemplo se ha determinado que los componentes complejos son los que implementan más de tres funcionalidades, en la Figura 4 se aprecia cómo el componente *CompPagoAnticipado* fue clasificado automáticamente por el razonador como complejo (*ComplexComponent*) porque implementa tres funcionalidades.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

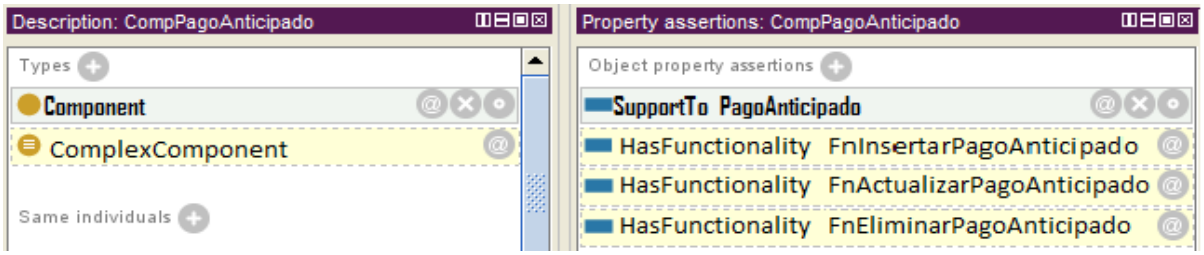

Figura 4. Inferencia de *ComplexComponent.*

Los componentes de los cuáles se consumen más servicios se van a clasificar como núcleos ya que un cambio en estos tiene un notable impacto en el sistema. Para ejemplificarlo se ha determinado que los componentes núcleos deben proveer como mínimo tres servicios que estén siendo consumidos por otros componentes. En la Figura 5 se puede apreciar que se ha clasificado automáticamente al componente *CompObligacionesPago* como núcleo porque provee tres servicios que están siendo consumidos.

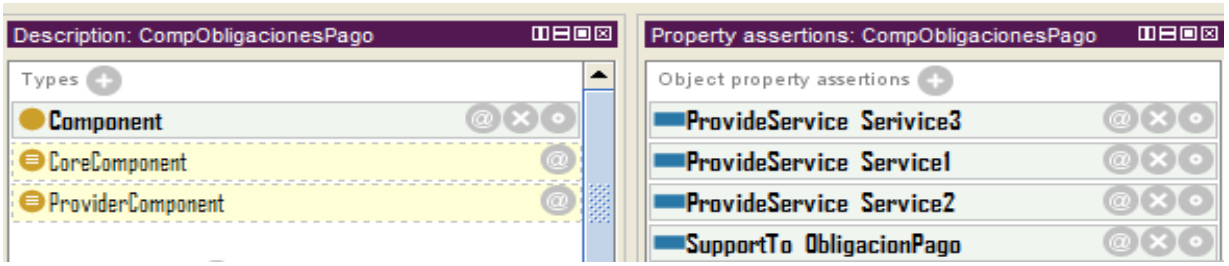

Figura 5. Inferencia de *CoreComponent.*

Validación

*Restricción R1***:** Para demostrar que la ontología es capaz de comprobar que no se viole la restricción R1 se estableció que el componente *BadDomainComponent* es un componente de dominio y que consume el servicio *Service1*. Dada estas condiciones el razonador clasifica automáticamente a *BadDomainComponent* como un *WrongComponent*, la Figura 6 muestra esa inferencia.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

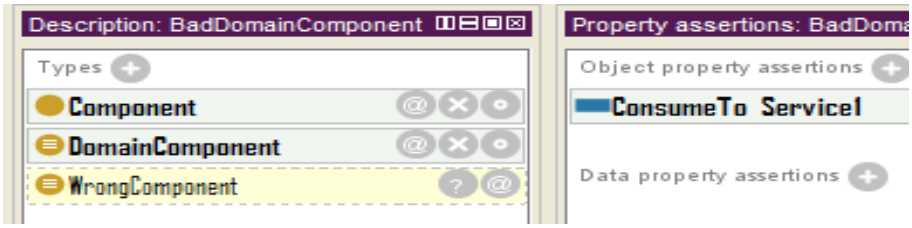

Figura 6. Inferencia de *WrongComponent.*

*Restricción R2***:** Para ejemplificar la verificación de las restricción R1 se especificó que el *BadDomainComponente* informatiza a los *DomainElement1* y *BusinessEelement1* que son elementos de dominio y patrimonial respectivamente. Al ejecutarse el razonador no encuentra un modelo donde un componente pueda informatizar un elemento patrimonial y un elemento de dominio a la vez, por lo que lanza la excepción que muestra la Figura 7.

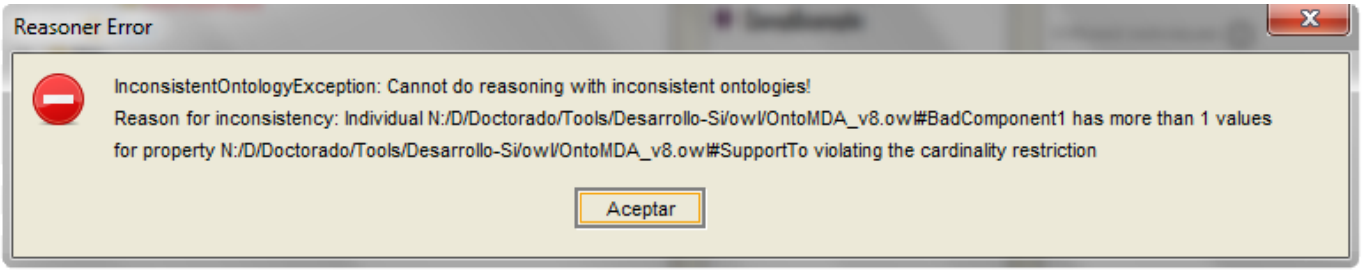

Figura 7. Excepción lanzada por violación de restricción R2.

*Restricción R3***:** Para ilustrar la verificación de la restricción R3 se especificó que *WrongFunctionality* es funcionalidad de *BadComponent1* y *BadComponent2.* Al activarse el razonador no encuentra un modelo donde dos componentes tengan implementada la misma funcionalidad, por eso lanza la excepción que se muestra en la Figura 8.

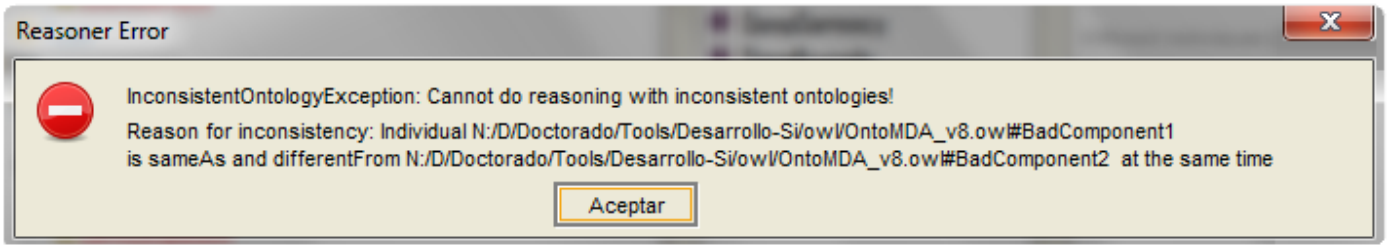

Figura 8. Excepción lanzada por violación de restricción R3.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 2, 2014
ISSN: 1994-1536 | RNPS: 0547
Pág. 85-101
http://rcci.uci.cu

En la ontología también se incluyen mecanismos para comprobar las restricciones R4 y R5, pero se omite su explicación para no extender este trabajo.

Comprobar manualmente que un modelo de componentes cumple con las restricciones mencionadas es una tarea muy compleja. Con este marco de trabajo, esta actividad se desarrolla de forma automatizada por lo que aumenta las probabilidades de detección de errores introducidos por personas que desarrollan el diseño arquitectónico. Este hecho evita que los errores se propaguen a otras fases, como la implementación, donde su solución es más costosa. La productividad durante la fase de diseño arquitectónico se favorece, porque la actividad de validación se realiza de manera automatizada y así los arquitectos se pueden dedicar a otras actividades del desarrollo de software. Para darle continuidad a esta investigación se debe aplicar un método empírico para constatar su impacto práctico.

Herramienta de soporte a las transformaciones de modelos

Se desarrolló un *plugin* para *Protégé* que ofrece soporte a la propuesta y prioriza la automatización en las transformaciones. La Figura 9 muestra la vista básica del plugin que consta de tres funcionalidades para ofrecer soporte a las transformaciones, por el alcance de este trabajo sólo se explica la funcionalidad *Exportar*.

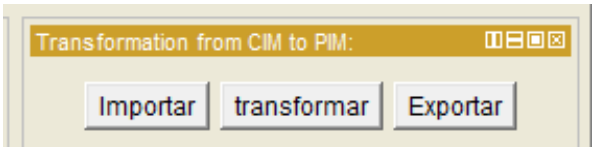

Figura 9. Funcionalidades del plugin.

La transformación de la ontología al modelo de componentes UML se desarrolla automáticamente. Al presionar el botón *Exportar* se crea un modelo de componentes UML (en formato XMI), en este modelo se diferencian las funcionalidades implementadas (se marcan con color verde) y por implementar (se marcan con color amarillo). Se han diferenciado los tipos de componentes, azules los de dominio, naranja los de negocio. En la Figura 10 se exhibe un ejemplo, se han ocultado algunos detalles para no sobrecargar la imagen.

Con el uso de esta herramienta se garantiza la automatización de las transformaciones, que es una de las prioridades de MDA. Aunque la propuesta está compuesta por dos modelos en el mismo nivel de abstracción los usuarios sólo tendrían que realizar manualmente la ontología, a partir de la cual se generará el modelo de componentes UML. Este elemento contribuye a aumentar la productividad durante el diseño arquitectónico pues los arquitectos solo tienen que realizar un modelo y el otro se genera de forma automatizada.

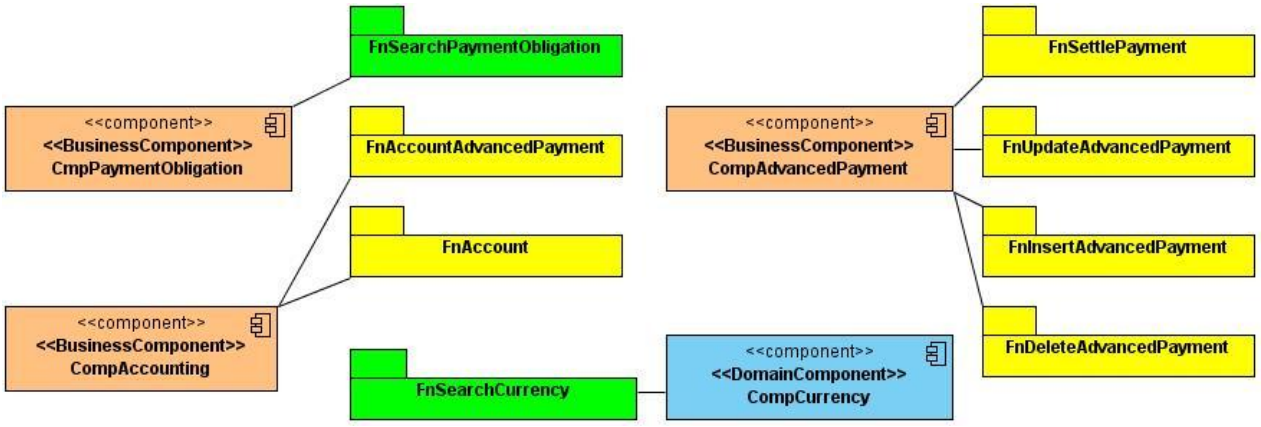

Figura 10. Modelo de componentes UML generado.

En el desarrollo de un sistema de gestión empresarial generalmente se involucra un número importante de personas. Específicamente en el área de arquitectura de sistema pueden existir varias personas realizando la misma actividad para diferentes partes del sistema. Por ejemplo, el diseño arquitectónico de alto nivel, donde se definen los componentes y sus interacciones para satisfacer los requerimientos deseados es realizado por varias personas. Aunque existen guías de cómo debe ejecutarse la actividad es muy difícil que el resultado sea homogéneo. En estas condiciones puede tener un efecto positivo el marco de trabajo descrito. Aquí se establecieron pautas que deben cumplir los modelos de componentes resultantes del diseño arquitectónico, además con el uso de una ontología se puede comprobar su cumplimiento.

Este marco de trabajo se puede enriquecer para que tenga en cuenta el nivel de los CIM. A partir de allí se podrán establecer mecanismos para que la actividad de diseño arquitectónico se realice de forma automática, partiendo de modelos que describan el negocio. El uso de ontologías y de los razonadores contribuirá a alcanzar esa meta.

El marco de trabajo se pudiera extender en el mismo nivel de los PIM. Es deseable que se consideren los modelos de datos y prototipos de interfaz de usuario que resultan de gran utilidad durante el proceso de desarrollo. Luego se podrían concebir modelos a nivel PSM que garanticen que todo el ciclo de vida sea cubierto por una aproximación MDA. De esta forma se podrán explotar en toda su magnitud las potencialidades de una propuesta MDA.

En el contexto MDA se promueve la automatización en todas las fases del desarrollo. El próximo paso como parte de esta investigación es obtener el modelo de componentes de manera automatizada. Para ello, se partirá del modelado de los procesos de negocio, que se explica en (Silega, 2014). Por lo tanto, como se desarrollará de forma automatizada permitirá que la lógica de diseño sea la misma pues se aplican las mismas reglas de transformación. Por otra parte,

este hecho también aumentará la productividad en el proceso de desarrollo pues las actividades de validación, entendimiento y transformación de los procesos de negocio se realizaría de forma automatizada lo que permitiría que los arquitectos se concentraran en otras actividades difíciles de automatizar.

Conclusiones

En esta investigación se presentan resultados teóricos a los que se arribó después de analizar las características del *framework* presentado. Se describió un marco de trabajo basado en MDA y complementado con ontologías que permite representar y validar modelos de componentes en el desarrollo de sistemas de gestión empresarial. La utilización de ontologías como parte del marco de trabajo descrito en OWL, el cual es un lenguaje formal, permite detectar con el apoyo de razonadores errores de forma automatizada en el modelo de componentes, como se evidenció en el chequeo de restricciones mostrado. Esto permite reducir la cantidad de errores en los modelos de componentes vinculados con la violación de las restricciones explicadas en el artículo.

El metamodelo de componentes como parte de MDA contribuye a aumentar la estandarización o uniformidad en la fase de diseño arquitectónico, ofreciendo conceptos específicos para elaborar los modelos de componentes. Como parte del metamodelo se establecen pautas de las relaciones entre los conceptos de cada modelo. Estos elementos reducen el impacto de la subjetividad de los arquitectos y aumenta la uniformidad entre diferentes modelos de componentes aunque sean realizados por distintas personas pues cuentan con los mismos elementos para el modelado.
La herramienta desarrollada permite que las personas que realizan la actividad de diseño arquitectónico sólo tengan que realizar manualmente un modelo, el otro se genera automáticamente, lo cual propicia que los arquitectos se concentren en otras actividades. Este aspecto contribuye a aumentar la productividad durante la construcción de la arquitectura de sistema y reducir el tiempo de esta fase, al igual que la detección automatizada de errores. Detectar y reducir el número de errores en la fase de construcción de arquitectura de sistema evita que esos errores se propaguen a otras fases del desarrollo, donde resultan más difíciles de detectar y su solución es más costosa.
Este *framework* constituye un paso importante en lograr una aproximación MDA que cubra todo el ciclo de vida del desarrollo de un sistema de gestión empresarial. Esto permitirá reducir la complejidad arbitraria como se plantea en (Selic, 2008), lo que constituye un objetivo primordial de la ingeniería de software como escribiera el destacado investigador Frederick Brooks en (Brooks, 1995).

Referencias

- BENVENUTO, A. *Implementación de sistemas ERP, su impacto en la gestión de la empresa e interacción con otras TIC*. CAPIV REVIEW, 2006, 4(1): p. 33-48.
- BLANCO, E. *Contabilidad y fiscalidad*. [en línea]. 2008. [Consultado el: 1 de noviembre de 2013]. Disponible en: [www.eumed.net/libros/2008b/396].
- BO, D. y LI-J. S. *Ontology-Based Model for Software Resources Interoperability*. Information Technology Journal, 2009, 8(6): p. 871-878.
- BOCANEGRA, J.; PEÑA, J., *et al*. *Una Aproximación MDA para Modelar Transacciones de Negocio a Nivel CIM*. En: Jornadas de Ingeniería del Software y Bases de Datos. España: 2008, p. 82-91.
- BROOKS, F. P. J. *The Mythical Man-Month*. North Carolina, ADDISON-WESLEY, 1995. 322 p.
- CARREÓN, M. C. *Construcción de un catálogo de patrones de requisitos funcionales para ERP*. Tesis de Máster en computación. Departamento de Lenguajes y Sistemas Informáticos. Universidad Politécnica de Catalunya, 2008.
- CHENGPU, L.; ROB, P., *et al*. *Ontology-Based Quality Attributes Prediction In Component-Based Development*. International Journal of Computer Science & Information Technology, 2010, 2(5): p. 12-29.
- CHUNGOORA, N. y YOUNG, R. I. M. *Ontology Mapping to Support Semantic Interoperability in Product Design and Manufacture*. En: Proceedings of the First International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08). Francia: 2008, p. 1-15.
- DE CASTRO, V.; M. E., *et al*. *Applying CIM-to-PIM Model Transformations for the Service-Oriented Development of Information Systems*. Information and Software Technology, 2010, 53(1): p. 87-105.
- GARLAN, D. *Formal Modeling and Analysis of Software Architecture: Components, Connectors, and Events*. En: Bernardo, M. y Inverardi, P. (editores). Formal Methods for Software Architectures. Pittsburgh, USA: Springer Berlin Heidelberg, 2003, p. 1-24.
- HORRIDGE, M. *A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.2*. The University Of Manchester, 2009, p. 108.
- KRUCHTEN, P. *An Ontology of Architectural Design Decisions in Software-Intensive Systems*. En Jan Bosch (editor). Proceedings of the 2nd Groningen Workshop on Software Variability Management. Groningen, NL: 2004, p. 1-8.

- LAGE, C.; SILEGA, N., *et al. Reglas contables. Componente para la contabilización de los procesos de una empresa en soluciones de software*. En: Taller internacional de Administración Financiera. Habana: 2009, p. 1-10.
- 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, 55(2): p. 89-200.
- MATOS, Y. y SILEGA, N. *Estilo arquitectónico para el sistema integrado de gestión Cedrux*. GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología, 2013,1(1): p. 1-12.
- MORA, B.; GARCÍA, F., et al. *Software Generic Measurement Framework Based on MDA*. IEEE Latin America Transactions, 2008, 6(4): p. 363-370.
- NOY, N. F. y MCGUINNESS, D. L. *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford: Stanford Medical Informatics, 2001, p. 25.
- OMG. *MDA Guide Version 1.0.1*. En: Miller, J. y Mukerji, J. (editores). OMG: 2003, p. 62.
- PAHL, C.; GIESECKE, S., *et al. Ontology-Based Modelling of Architectural Styles*. Information and Software Technology, 2009, 51(12): p. 1739-1749.
- SÁNCHEZ, M. Á.; FERMOSO, 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*. PNIS, 2008: p. 64-71.
- SELIC, B. *Manifestaciones sobre MDA*. Novática: Revista de la Asociación de Técnicos de la Informática, 2008, 192: p. 13-16.
- SILEGA, N.; LOUREIRO, T; NOGUERA, M. *Marco de trabajo dirigido por modelos y basado en ontologías para la descripción y validación semántica de procesos de negocio*. IEEE Latin America Transactions, 2014, 12(12), p. 292-299.
- SINGH, Y. y SOOD, M. *The Impact of the Computational Independent Model for Enterprise Information System Development*. International Journal of Computer Applications, 2010, 11(8): p. 21-26.
- TAYLOR, R.; MEDVIDOVIC, N., *et al*. *Software Architecture: Foundations, Theory and Practice*. En: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. Cape Town: ACM, 2009, p. 471-472.

- TETLOW, P.; PAN, J.; OBERLE, D.; WALLACE, E.; USCHOLD, M; y KENDALL, E. *Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering*. W3C Working Draft Working Group Note 2006/02/11, W3C, 03 2006. Disponible en: http://www.w3.org/2001/sw/BestPractices/SE/ODA/.
- XING, J. y AH-HWEE, T. *CRCTOL: A semantic-based domain ontology learning system*. Journal of the American Society for Information Science & Technology, 2010, 61(1): p. 150-168.