Análisis del impacto de una solución tecnológica de apoyo al diseño arquitectónico en proyectos de software

Impact analysis of a technological solution to support the architecture design in software projects

Nemury Silega Martínez

Danaysa Macías Hernández

Universidad de las Ciencias Informáticas. La Habana. Cuba.

Universidad de Matanzas "Camilo Cienfuegos". Matanzas. Cuba

Resumen

La fase de diseño arquitectónico es crucial para el éxito de un proyecto de software. En esta fase se definen los componentes de alto nivel que incluirá el sistema, así como sus relaciones. Usualmente el diseño arquitectónico se desarrolla de forma intuitiva por los arquitectos por lo que la calidad del diseño resultante dependerá de sus habilidades. No obstante, es común que se generen errores durante esta fase. Estos errores si no son detectados a tiempo se propagan a fases posteriores donde su solución demanda más tiempo y esfuerzo provocando desviaciones en los objetivos del proyecto. Este artículo presenta una solución tecnológica de apoyo al diseño arquitectónico. Esta solución es basada en el Desarrollo Dirigido por Modelos (MDD). Además, se analizan evidencias cuantitativas que demuestran el impacto favorable en la reducción de errores y el del tiempo empleado para el diseño arquitectónico.

Palabras clave: herramienta, impacto, MDD, diseño arquitectónico

Abstract

The architecture design phase is crucial to the success of a software project. In this phase, the high-level components that the system will include as well as its relationships are defined. Usually the architecture design is developed intuitively by the architects so the quality of the resulting design will depend on their skills of the architects. However, it is common to find errors that are generated during this phase. These errors if not

detected in time are propagated to later phases where their solution demands more time and effort causing deviations in the project objectives. In this article we present a technological solution to support the architecture design. This tool is based on Model Driven Development (MDD). We also carry out an analysis with quantitative evidences that show the favorable impact in the reduction of errors and the time used for the architectural design.

Keywords: *tool, impact, MDD, architecture design*

Introducción

Frederick Brooks en su libro "The Mythical Man-Month" plantea que el desarrollo de software es un proceso complejo. Esta complejidad se puede analizar en dos dimensiones: esencial y arbitraria. La complejidad esencial es inherente al proceso (ejemplo: procesos de gestión empresarial, procesos de gestión hospitalaria, entre otros). Esta complejidad es inevitable, pues no depende del proceso de desarrollo de software (Selic, 2008).

Por otra parte, la complejidad arbitraria se refiere al proceso interno de desarrollo de software y tiene en cuenta las tecnologías, capacidad de los desarrolladores, factores objetivos y subjetivos. Esta complejidad es la que más afecta al proceso de desarrollo y a menudo ocasiona pérdida de tiempo en resolver errores provocados por detalles de la tecnología empleada. Por ejemplo, declarar mal una variable podría causar un error que consuma varios días detectarlo o el cambio de una plataforma podría implicar meses de capacitación y de actualización del código.

En la fase de desarrollo del diseño arquitectónico o diseño conceptual se identifican los componentes a un alto nivel de abstracción que son necesarios para satisfacer los requisitos según las demandas de los clientes (Al-Jamimi & Ahmed, 2013). Las decisiones que se toman durante el diseño tienen una influencia considerable en el resto de las fases del desarrollo (Al-Jamimi & Ahmed, 2013). Las características de los sistemas complejos, por ejemplo los sistemas de Gestión empresarial (SGE), determinan que la fase de diseño arquitectónico tenga una alta complejidad esencial, lo que resalta la necesidad de reducir la complejidad arbitraria en esta fase, donde los arquitectos emplean una cantidad significativa de tiempo y esfuerzo (Al-Jamimi & Ahmed, 2013).

Es común que la fase de diseño arquitectónico se realice de manera intuitiva y dependiendo de la pericia y creatividad de los arquitectos (Al-Jamimi & Ahmed, 2013) sin seguir una guía formal ni una vía efectiva de comprobar la calidad del diseño resultante. Esta situación determina frecuentes rediseños para solucionar problemas detectados en fases posteriores y que no se detectaron en el diseño inicial. Las modificaciones que provoca el rediseño suele provocar un impacto negativo en el código. Esto implica pérdida de tiempo, reducción de la calidad y eleva el costo del desarrollo. Varios estudios demuestran que los errores post-implementación cuestan 100 veces más que los errores que se producen durante la fase de diseño (Boehm, 1981; Moreno, Snoeck, Reijers, & Rodríguez, 2014; Sánchez, García, Ruiz, & Mendling, 2012).

Barjis (2008) indaga sobre los problemas en el diseño durante el desarrollo de software, en especial para sistemas de alta complejidad como los SGE. En el estudio se concluye que el pobre modelado de los procesos de negocio y su correspondencia con el diseño es un factor determinante en el fallo de los

sistemas. Otros estudios afirman que es común la presencia de errores en los modelos de procesos de negocio (Mendling, 2009; Moreno et al., 2014). El deficiente modelado de los procesos de negocio influye en los problemas mencionados. La descripción de procesos de negocio es el artefacto base para el diseño arquitectónico de un SGE. Sin embargo, es común que este artefacto no sea validado correctamente y sus defectos se propaguen al diseño arquitectónico. Como reconocen diversos estudios, esto aumenta exponencialmente el costo y esfuerzo de la solución de los errores (Moreno et al., 2014; Sánchez et al., 2012; Sánchez, Ruiz, García, & Piattini, 2013; Wand & Weber, 2002).

Cuando varias personas describen los procesos de negocio se obtienen descripciones heterogéneas que dificultan la comprensión de los que realizan el diseño arquitectónico (Endert, Kuster, Hirsch, & Albayrak, 2007). Como conclusión del diagnóstico preliminar se afirma que estos elementos afectan a la calidad de la descripción de los procesos de negocio, así como al tiempo que se emplea para su desarrollo e influyen negativamente en el proceso de diseño arquitectónico.

En un sistema de alta complejidad el diseño arquitectónico es realizado por diferentes personas (Endert et al., 2007). Esto propicia, al igual que en la descripción de procesos de negocio, heterogeneidad en el resultado. La heterogeneidad en el diseño dificulta la identificación de modificaciones necesarias para adaptar el sistema a cambios en los procesos. Como la relación entre los elementos de un proceso y los elementos del diseño no está bien identificada, es difícil evaluar el impacto de un cambio en un proceso. Si el diseño no es estándar resulta difícil comprender su lógica y la capacitación de nuevo personal para esta actividad suele ser lenta y poco productiva.

Como instrumento para abordar la alta complejidad arbitraria en el desarrollo de software surge el paradigma de Desarrollo dirigido por modelos (MDD, por sus siglas en inglés). En la literatura se reconoce la utilidad de adoptar propuestas basadas en MDD para el desarrollo de sistemas de alta complejidad, debido a la importancia de aumentar el nivel de abstracción para mejorar la comprensión, comunicación y análisis por los humanos (Mohagheghi et al., 2011).

Arquitectura dirigida por modelos (MDA, por sus siglas en inglés) es la propuesta de mayor relevancia dentro de los enfoques que siguen el MDD. MDA organiza el desarrollo a partir de tres niveles de abstracción: modelos independientes de la computación (CIM), modelos independientes de la plataforma (PIM) y modelos específicos de la plataforma (PSM).

Considerando todos los elementos planteados anteriormente, el objetivo de este trabajo es describir una solución tecnológica que apoye el proceso de diseño arquitectónico basado en MDA. Esta propuesta sirve de base tecnológica a un método presentado anteriormente con el propósito de reducir la complejidad arbitraria del diseño arquitectónico. Un elemento clave del método es que explota las ontologías para describir y validar los modelos. En consecuencia, la herramienta presentada permite la manipulación y transformación de ontologías. Finalmente se presentan evidencias cuantitativas del impacto de esta solución en el desarrollo del diseño arquitectónico.

Materiales y métodos

Arquitectura Dirigida por Modelos (MDA)

MDA es una propuesta promovida por el Object Management Group (OMG, por sus siglas en inglés). Dentro de las ideas fundamentales de este paradigma 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(OMG, 2003).

MDA pretende obtener aplicaciones con alta flexibilidad en la implementación, integración, mantenimiento y prueba. 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 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. El modelo resultante puede 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 automatizada garantizando la calidad de los modelos.

En varias propuestas se demuestra el impacto positivo de la aplicación de aproximaciones MDA en el desarrollo de software (Bocanegra, Peña, & Ruiz Cortés, 2008), (Singh & Sood, 2010), (De Castro, Marcos, & Vara, 2010), (Sánchez Vidales, Fermoso García, & joyanes Aguilar, 2008) y (Mora et al., 2008). Un estudio realizado permitió analizar las evidencias empíricas del efecto de la aplicación de enfoques MDA en el desarrollo de software (Y. Martínez, Cachero, & Meliá Beigbeder, 2011). 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. Un experimento para comparar la aceptación del desarrollo dirigido por modelos con métodos tradicionales demostró el creciente interés en los métodos de desarrollo dirigido por modelos (Yulkeidi Martínez, Cachero, & Meliá, 2012). La consideración de estas propuestas determinó que en este trabajo se adoptara MDA como base de la propuesta.

Ontologías

Una ontología se define 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 (Noy & McGuinness, 2001. 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 el surgimiento de la 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 (Resource Description Framework), RDF Schema (o RDF-S), OWL (Xing & Ah-Hwee) y otros. Dentro de todos, se distingue OWL por las facilidades que ofrece, en especial 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, determinaron que en esta propuesta se asuma OWL como el lenguaje para la representación de ontologías.

Varias propuestas demuestran las potencialidades de las ontologías en la ingeniería del software, en especial para la descripción y validación de las arquitectura de software (Pahl, Giesecke, & Hasselbring, 2009), (Bo & Li-juan, 2009),(Chengpu, Rob, & Xiaodong, 2010), (Chungoora & Young, 2008) y (Kruchten, 2004). Estos beneficios determinaron que en la propuesta que se presenta en este artículo se adoptaran las ontologías en OWL para la especificación de modelos.

Método basado en MDA y ontologías para el diseño arquitectónico

A continuación se describe brevemente el método de diseño arquitectónico (Silega, Noguera, & Macias, 2016), la Figura 1 muestra una vista general del método. Los rectángulos representan los tipos de modelos y las flechas las transformaciones entre modelos. La propuesta incluye tres tipos de modelos independientes de la computación y dos tipos de modelos independientes de la plataforma, así como, cuatro etapas de transformaciones entre modelos, para ir de unos tipos de modelos a otros. En trabajos previos se describieron y justificaron los CIMs y los PIMs que componen el método (Silega, Loureiro, & Noguera, 2014; Silega, Macías, Matos, & Febles, 2014).

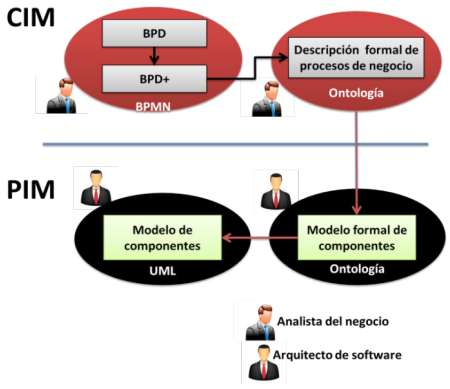

Figura 1. Vista general de la propuesta: modelos y transformaciones.

Modelos Independientes de la computación

BPD: Permite describir los procesos de negocio de manera que puedan ser entendidos tanto por los especialistas del negocio como por los desarrolladores de software. Sin embargo, ciertos conceptos identificados de la gestión empresarial no pueden representarse explícitamente en un BPD lo que motiva el siguiente modelo independiente de la computación.

BPD+: Es un tipo de modelo intermedio para enriquecer el BPD con información propia del dominio. Aquí se especifican explícitamente los conceptos identificados sobre el dominio.

Modelo ontológico: La ontología contiene clases, relaciones y restricciones para representar un proceso de negocio. Algunas de las clases son: *Actividad, Proceso, ElementoFlujo, Evento, Compuerta, Estado* y *Objeto* que se asocian directamente con conceptos de BPMN. En la ontología también se incluyen propiedades para vincular las actividades con los elementos (*patrimoniales* o de *dominio*) y los tipos de acciones que se pueden realizar.

Modelos Independientes de la Plataforma (PIM)

Para obtener una vista del sistema independiente de la plataforma esta propuesta se centra en la arquitectura de sistema. Esta provee una vista de alto nivel de abstracción, muestra una división del sistema en componentes y las interacciones que ocurren entre ellos. Su propósito es mostrar cómo los componentes del sistema se interrelacionan para satisfacer las funcionalidades identificadas en el modelado de los procesos de negocio. Sus tres elementos principales son: Componente, Servicios y Funcionalidad.

Modelo de componentes ontológico: En la ontología se definen los mecanismos necesarios para describir los conceptos relacionados con la arquitectura de sistema. Se definen como clases *Componentes, Servicios y Funcionalidades*. Se incluyen propiedades que relacionan los conceptos del modelo. Por ejemplo, se especifica que un *Servicio* es provisto por algún *Componente* y que un *Componente* implementa ciertas *Funcionalidades*. Además, se realiza una clasificación de los componentes según su función en: *Componentes de negocio*, *Componentes del dominio* y *Componentes tecnológicos*.

Modelo de componentes UML: Tener expresado el modelo de componentes en una ontología genera los beneficios que se describieron anteriormente. Sin embargo, las otologías están diseñadas para comunicar conocimiento entre computadoras por lo que suele resultar difícil analizar por los humanos. Este hecho motiva que también se represente el modelo de componentes en una notación que facilite la comprensión por los humanos. En este caso se emplea el diagrama de componentes definido en UML. Se seleccionó este tipo de modelo porque forma parte del estándar UML el cual es ampliamente conocido en la comunidad de desarrolladores de software y puede ser interpretado por diversas herramientas. Por lo tanto, como muestra la Figura 1, el nivel PIM está compuesto por un modelo de componentes formal expresado mediante una ontología y un modelo de componentes UML.

Resultados y discusión

A continuación, se describe la herramienta desarrollado con el fin de ofrecer soporte tecnológico al método presentado anteriormente. La herramienta se desarrolló en forma de plugin para Protégé e informatiza las transformaciones definidas en el método. Antes de describir las funcionalidades de la herramienta primero se describen brevemente las trasformaciones que concibe el método de diseño arquitectónico:

Transformación 1 (BPD -> BPD+): En este paso se enriquece el BPD con la información relacionada con los conceptos propios de la gestión empresarial ya identificados.

Transformación 2 (BPD+ -> Modelo de proceso formal (Ontología)): Esta transformación permite obtener automáticamente una ontología con la descripción de un proceso de negocio expresado en un BPD. Se definieron las reglas de transformación que permite relacionar los elementos entre ambos modelos.

Transformación 3 (Modelo de proceso formal (Ontología) -> Modelo de componentes formal (Ontología)): Esta transformación permite obtener un PIM a partir de un CIM. En este caso se obtiene un modelo de componentes a partir de un modelo de procesos de negocios.

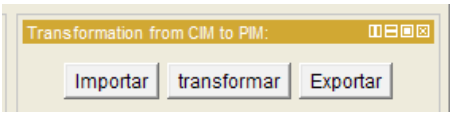

Figura 2. Funcionalidades del plugin.

La Figura 2 muestra la vista básica del plugin que consta de tres funcionalidades. Con la funcionalidad *importar* se ejecuta la transformación 1 y la transformación 2. Mientras que las funcionalidades *transformar* y *exportar* soportan las transformaciones 3 y 4 respectivamente. Para explicar mejor las funcionalidades de la herramienta se irá ilustrando con un caso de estudio donde se ejecuta la cadena de transformaciones definidas en el método. En este caso de estudio se parte del BPD referente al proceso *Liquidar pagos anticipados en una empresa*. El flujo comienza cuando el financista efectúa la revisión del expediente del proveedor que debe realizar el pago anticipado. Considerando las facturas que obedecen al pago anticipado y teniendo en cuenta los datos de la compra, este coteja las facturas con el pago anticipado. Luego el financista actualiza el expediente del proveedor y finalmente envía a contabilizar la operación realizada. La Figura 3 muestra el BPD correspondiente a ese proceso.

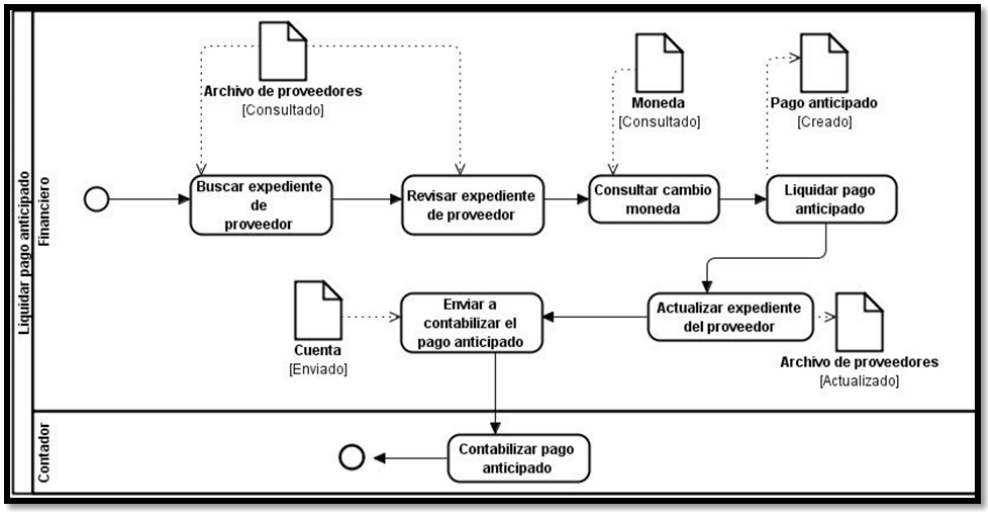

Figura 3. BPD del proceso Liquidar pago anticipado. Fuente: Elaboración propia.

Cuando se selecciona la funcionalidad *importar* de la vista básica (Figura 2) de la herramienta se mostrará la interfaz que se muestra en la Figura 4. Como se puede apreciar, el primer paso es definir la dirección del modelo de procesos de negocios expresado en XMI que se desea transformar, lo que permitirá cargar automáticamente todas sus actividades para que se incorpore la información específica del dominio. En este caso se muestran las actividades del proceso mostrado en la figura 3. Luego de completar este paso se podrá transformar el modelo automáticamente a una ontología describiendo el proceso. La figura 5 muestra las actividades creadas en la ontología luego de la transformación.

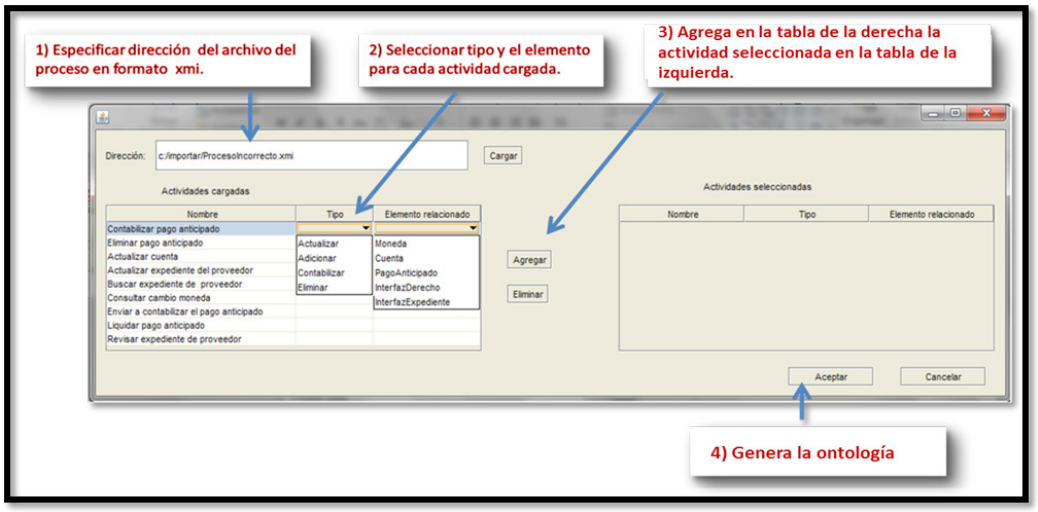

Figura 4. Ventana para importar elementos del BPD.

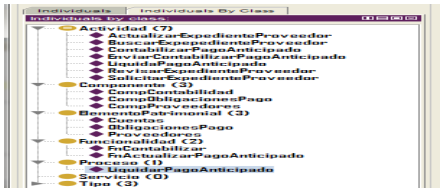

Figura 5. Actividades del proceso en la Ontología.

Luego de generar el modelo de procesos de negocio especificado en la ontología se puede transformar a un modelo de componentes (transformación 3). Para ejecutar esta transformación se utiliza la funcionalidad *transformar* de la herramienta. Cuando se presione el botón transformar se genera automáticamente el modelo de componentes siguiendo las reglas especificadas. La figura 6 muestra los componentes y funcionalidades que se generan después de ejecutar la trasformación.

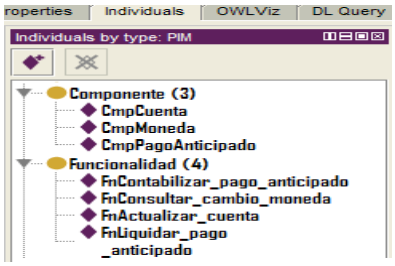

Figura 6. Componentes y Funcionalidades generadas.

Una vez que se cuenta con el modelo de componentes expresado en la ontología corresponde transformarlo al modelo de componentes UML (Transformación 4). Al presionar el botón exportar en la herramienta se genera automáticamente un modelo de componentes UML (en formato XMI). En la Figura 7 se exhibe el modelo generado a partir del modelo de componentes expresado en la ontología. En el modelo todas las funcionalidades son amarillas lo que significa que no están implementadas. *CmpMoneda* es azul porque es un componente de dominio, los componentes *CmpCuenta* y *CmpPagoAnticipado* son rojos porque son de negocio. Esta diferencia de colores en dependencia de si está implementada la funcionalidad o del tipo de componente podría facilitar el análisis de los involucrados en el diseño arquitectónico.

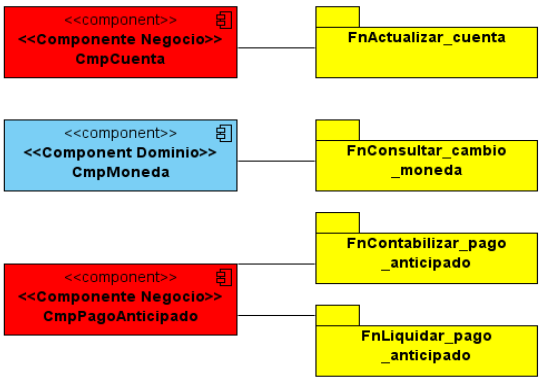

Figura 7. Modelo de componentes UML generado.

Este caso de estudio ha permitido ilustrar cómo la herramienta facilita la automatización de las transformaciones, una prioridad de MDA. Aunque la propuesta está compuesta por cinco modelos en diferentes niveles de abstracción, los usuarios sólo deben realizar manualmente el BPD a partir del cual se generan los otros modelos.

Evaluación del impacto de la propuesta

Para comprobar el impacto de la propuesta en la reducción de la cantidad de errores en los modelos de componentes se realizó un cuasi experimento con integrantes de un proyecto para el desarrollo de un sistema de gestión empresarial. Participaron integrantes del proyecto que hubiesen realizado la actividad de diseño arquitectónico y que tuvieran experiencia en el desarrollo de software para realizar una valoración

de la complejidad del proceso. El cuasi experimento se desarrolló con 11 arquitectos o programadores de experiencia. Todos los participantes son ingenieros informáticos y tienen al menos dos años de experiencia en el desarrollo de SGE.

Se les entregó a los participantes el diagrama de procesos de negocio del proceso *Liquidar pago anticipado* del área de Finanzas. Se les orientó que realizaran una propuesta de diseño arquitectónico a partir de las pautas definidas en el proyecto y de su experiencia personal. En la Tabla 1 se muestra el resultado de la primera medición realizada sin aplicar la herramienta. Como se puede apreciar en todos los casos se introdujeron más de dos errores y se obtuvo un promedio de 4.27 errores por cada arquitecto. Los errores detectados obedecen fundamentalmente a las siguientes clasificaciones:

- Incapacidad del diseño presentado para abarcar todas las actividades del proceso de negocio, lo que implica que el sistema resultante no apoye totalmente los procesos del negocio que estaban previstos.
- Identificación de componentes sin sentido para el negocio, lo que suele generar innecesarias integraciones afectando el rendimiento del sistema y la reutilización.
- Integración incorrecta según las pautas establecidas en el proyecto para los diferentes tipos de componentes, lo que puede afectar la reutilización y generar dependencias no deseadas entre los componentes alcanzando un alto acoplamiento. Aquí se aprecia la violación del principio de diseño referente a mantener un bajo acoplamiento entre los componentes (Larman, 1999).
- Excesiva responsabilidad para un componente, lo que ocasiona baja cohesión. Igualmente aquí se aprecia la violación de uno de los principios fundamentales del diseño referente a la alta cohesión (Larman, 1999).

Tabla 1. Resultado del cuasi experimento con los arquitectos.

Participante	Cantidad errores sin la propuesta	Participante	Cantidad errores sin la propuesta
A1	7	**A7**	3
A2	5	**A8**	3
A3	2	**A9**	5
A4	2	**A10**	7
A5	5	**A11**	2
A6	6		

Con la utilización de la herramienta se redujo a cero el número de errores introducidos durante el diseño en todos los casos, demostrando que su uso tiene un efecto positivo. Estos resultados obedecen en gran medida a las potencialidades que ofrecen las ontologías y la transformación automatizada de modelos.

El cuasi experimento también permitió comprobar el impacto positivo de la propuesta para lograr homogeneizar la lógica de diseño. De los modelos de componentes propuestos por los participantes, sólo dos poseen un alto grado de similitud, influenciado en gran medida porque son programadores del mismo subsistema y que han estado más de tres años trabajando juntos. Como se esperaba, con la herramienta siempre se generó la misma solución lo que permite concluir que la propuesta asegura la homogeneidad en la lógica de diseño.

Conclusiones

En el documento se describió una solución que sirve de marco tecnológico a un método de diseño arquitectónico basado en MDA. Se ilustró cómo las funcionalidades de la herramienta permiten ejecutar de forma automatizada las transformaciones definidas en el método. Además, evita que los arquitectos tengan que crear manualmente lo modelos ontológicos con los que generalmente no están familiarizados. Con esta solución se explotan las promesas de los enfoques MDA. Se ofrecieron evidencias cuantitativas del impacto favorable del uso de la herramienta en la reducción de errores durante el diseño arquitectónico, así como la mejora en la homogeneización de los artefactos generados. Con el empleo de esta herramienta también se podría contribuir a reducir el tiempo de desarrollo del diseño arquitectónico. Actualmente se avanza en la extensión de la propuesta para que abarque otros modelos en diferentes niveles de abstracción de manera que se apoye no solo el trabajo de los arquitectos.

Referencias

Al-Jamimi, H., & Ahmed, M. (2013). Transition from Analysis to Software Design: A Review and New Perspective. *International Journal of Soft Computing and Software Engineering [JSCSE], 3*(3), 169-176.

Barjis, J. (2008). The importance of business process modeling in software systems design. *Science of Computer Programming, 71*(1), 73-87.

Bo, D., & Li-juan, S. (2009). Ontology-Based Model for Software Resources Interoperability. *Information Technology Journal, 8*(6), 871-878.

Bocanegra, J., Peña, J., & Ruiz Cortés, A. (2008). *Una Aproximación MDA para Modelar Transacciones de Negocio a Nivel CIM.* Paper presented at the Jornadas de Ingeniería del Software y Bases de Datos, España.

Boehm, B. W. (1981). *Software Engineering Economics*: Prentice-Hall, Englewood Cliffs.

Chengpu, L., Rob, P., & Xiaodong, L. (2010). ONTOLOGY-BASED QUALITY ATTRIBUTES PREDICTION IN COMPONENT-BASED DEVELOPMENT. *International Journal of Computer Science & Information Technology, 2*(5), 12-29.

Chungoora, N., & Young, R. I. M. (2008). *Ontology Mapping to Support Semantic Interoperability in Product Design and Manufacture.* Paper presented at the MDISIS 2008.

De Castro, V., Marcos, E., & Vara, J. M. (2010). Applying CIM-to-PIM model transformations for the service-oriented development of information systems. *Information and Software Technology, 53*(1), 87-105.

Endert, H., Kuster, T., Hirsch, B., & Albayrak, S. (2007). Towards a Mapping from BPMN to Agents. In M. Baldoni, C. Baroglio & V. Mascardi (Eds.), *Agent, Web Services, and Ontologies Integrated Methodologies* (Vol. 4504, pp. 92-106): Springer.

Kruchten, P. (2004). *An Ontology of Architectural Design Decisions in Software-Intensive Systems*. Vancouver, B.C., Canada . University of British Columbia.

Larman, C. (1999). *UML Y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado* (1 ed.). México: Prentice Hall.

Martínez, Y., Cachero, C., & Meliá Beigbeder, S. (2011). Evidencia empírica sobre mejoras en productividad y calidad en enfoques MDD: un mapeo sistemático. *REICIS Revista Española de Innovación, Calidad e Ingeniería del Software*

7(2), 6-27.

Martínez, Y., Cachero, C., & Meliá, S. (2012). MDD vs. Traditional Software Development: a practitioners subjective perspective. *Information and Software Technology*(0).

Mendling, J. (2009). Empirical Studies in Process Model Verification. In K. Jensen & W. M. P. van der Aalst (Eds.), *Transactions on Petri Nets and Other Models of Concurrency II* (Vol. 5460, pp. 208-224): Springer Berlin Heidelberg.

Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M. A., Nordmoen, B., & Fritzsche, M. (2011). Where does model-driven engineering help? Experiences from three industrial cases. *Software & Systems Modeling*.

Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., . . . Ramos, I. (2008). Software generic measurement framework based on MDA. *IEEE Latin America Transactions, VOL. 6,*(4).

Moreno, I., Snoeck, M., Reijers, H. A., & Rodríguez, A. (2014). A systematic literature review of studies on business process modeling quality. *Information and Software Technology*.

OMG. (2003). *MDA Guide Version 1.0.1*.

Pahl, C., Giesecke, S., & Hasselbring, W. (2009). Ontology-based modelling of architectural styles. *Information and Software Technology, 51*(12), 1739-1749.

Sánchez, L., García, F., Ruiz, F., & Mendling, J. (2012). Quality indicators for business process models from a gateway complexity perspective. *Information and Software Technology, 54*(11), 1159-1174.

Sánchez, L., Ruiz, F., García, F., & Piattini, M. (2013). Improving Quality of Business Process Models. In L. Maciaszek & K. Zhang (Eds.), *Evaluation of Novel Approaches to Software Engineering* (Vol. 275, pp. 130-144): Springer Berlin Heidelberg.

Sánchez Vidales, M. Á., Fermoso García, A., & joyanes Aguilar, L. (2008). 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.

Selic, B. (2008). Manifestaciones sobre MDA. *Novática*(192), 13-16.

Silega, N., Loureiro, T. T., & Noguera, M. (2014). Marco de trabajo dirigido por modelos y basado en ontologías para la descripción y validación semántica de procesos de negocio. *Latin America Transactions, IEEE (Revista IEEE America Latina), 12*(2), 292-299.

Silega, N., Macías, D., Matos, Y., & Febles, J. P. (2014). Framework basado en MDA y ontologías para la representación y validación de modelos de componentes. *Revista Cubana de Ciencias Informáticas (RCCI), 8*(2), 85-101.

Silega, N., Noguera, M., & Macias, D. (2016). Ontology-based Transformation from CIM to PIM. *IEEE Latin America Transactions, 14*(9), 4156-4165.

Singh, Y., & Sood, M. (2010). The Impact of the Computational Independent Model for Enterprise Information System Development. *International Journal of Computer Applications, 11*(8).

Wand, Y., & Weber, R. (2002). Research commentary: information systems and conceptual modeling - a research agenda. *Information Systems Research, 13*(4), 363-376.