Extendiendo Transformaciones MDA con Metamodelo de Patrones de Diseño

Luis Roqué Fourcade*, Liliana Arakaki*, Daniel Riesco*, Germán Montejano*, Narayan Debnath†
*Dpto. de Informática - Universidad Nacional de San Luis – E. de Los Andes 950, San Luis, Argentina
{araroq, liliana.arakaki}@yahoo.com, {driesco, gmonte}@unsl.edu.ar

†Computer Science Department - Winona State University
Winona, MN 55987, USA
ndebnath@winona.edu

Resumen

Object Management Group (OMG) ha producido un importante paso hacia el modelado del Proceso de Desarrollo de Software (PDS), el enfoque Model Driven Architecture (MDA). Sin embargo, no ha tenido el impacto esperado en la calidad de las instancias del modelo ni en los productos resultantes. La principal razón radica en el nivel de abstracción alejado de los dominios para los cuales deben generarse dichas instancias. En el caso de Diseño de Software (DS), no incluye abstracciones del proceso como tal y mucho menos de la actividad principal, la toma de decisiones. Presentamos aquí una línea de investigación que propone extender el modelo de transformaciones MDA con abstracciones inherentes a la actividad de síntesis y evaluación de alternativas para la toma de decisiones. Proponemos utilizar Patrones de Diseño de Software como el estándar de representación este tipo de conocimiento e incluir un metamodelo del mismo, para posibilitar su inyección en transformaciones MDA. Estas extensiones permitirán la generación de instancias del PDS, más cercanas a los dominios tecnológicos y de aplicación direccionados por los patrones, incrementando así el perfil ingenieril de la actividad.

Palabras clave: MDA, Arquitectura de Software, Model Driven Development, Model Driven Architecture, Proceso de Desarrollo de Software.

Contexto

El presente trabajo se desarrolla en el marco del proyecto 'Ingeniería de software: aspectos de alta sensibilidad en el ejercicio de la profesión de ingeniero de software', de la Fac. de Cs. Físico- Matemáticas y Naturales de la Universidad Nacional de San Luis (UNSL), bajo el número 22F222.

1. Introducción

La Ingeniería de Software ha producido avances realmente significativos a lo largo de su proceso evolutivo. Entre estos, la tecnología de componentes, la evolución arquitectural, la reutilización de especificaciones de diseño y el modelado del PDS, donde destaca el enfoque MDA.

Los resultados benéficos se verifican a través de diferentes modelos aplicables al desarrollo de software. Sin embargo, existe aún un vacío importante en cuanto a la asimilación de estos resultados en modelos del PDS que impacten positivamente en el perfil ingenieril de la actividad. En el área de DS, la Ingeniería de Software provee un soporte amplio para la especificación de todos los artefactos resultantes de actividades del proceso, pero no ocurre lo mismo con el proceso propiamente dicho y mucho menos con su actividad característica y repetitiva: síntesis y evaluación de alternativas para la toma de decisiones. Entre las principales consecuencias que se derivan de este bajo perfil ingenieril, se destaca el pobre diseño arquitectural que resulta en arquitecturas complejas, con alto costo de mantenimiento y que se degradan significativamente durante su evolución.

El enfoque MDA proporciona un modelo del PDS que actúa como una meta-arquitectura que integra avances en las diferentes dimensiones mediante la contribución de metamodelos en el marco del estándar. El modelo incluye:

- ✓ Una arquitectura de metamodelado basada en MetaObject Facility (MOF) [1], un metalenguaje basado en una simplificación de las capacidades de modelado de clases de UML2 más un núcleo de capacidades para gestión de metamodelos.
- ✓ Una arquitectura del PDS que ordena el proceso en niveles de abstracción distribuidos entre modelos de negocio y de implementación
- ✓ Un modelo de transformación a nivel de metamodelos que permite evolucionar en el PDS a través de los niveles de abstracción

Este modelo permite contar con procesos de desarrollo con capacidad de soportar la actividad de desarrollo a partir del conocimiento contenido en sus abstracciones. Sin embargo, el nivel de abstracción

propuesto se encuentra aún demasiado alejado de los dominios tecnológicos y de aplicación para los cuales se pretende desarrollar las aplicaciones. Así, las instancias de PDS construidas, no incluyen instancias de abstracciones, incluidas en la meta-arquitectura, que garanticen un soporte efectivo a la actividad de desarrollo. Claramente, si las incluyese, permitiría construir instancias con una capacidad de soporte que elevaría significativamente el perfil ingenieril de la actividad [2]. En este sentido, en el año 2016 en [3], presentamos una línea de investigación que propuso definir la extensión de la meta-arquitectura aportada por el enfoque MDA. Su desarrollo durante 2016 derivó en una publicación con una propuesta de extensión de la meta-arquitectura [2] que incluyó:

- ✓ Una definición de DS que sintetiza el universo disponible e incorpora el enfoque de patrones adicionando estructura tanto a la definición del proceso como del artefacto resultante, la Arquitectura de Software.
- ✓ Un modelo extendido de transformaciones MDA que, propone la inyección del conocimiento representado por patrones en las ejecuciones de transformaciones.

1.1. Trabajos relacionados

Muchos trabajos de investigación enfocan desde el diseño arquitectural el problema e intentan aportar soluciones que mejoren el proceso por el cual se obtienen y mantienen las arquitecturas de software y, la mayoría lo hace sin considerar el PDS.

Dada la naturaleza del proceso de diseño, el razonamiento o la base lógica del curso de acciones asociado a la actividad de toma de decisiones se ha convertido en el hilo conductor de buena parte de estos trabajos. La mayoría de sus autores coinciden en diagnosticar el bajo perfil ingenieril que caracteriza actualmente al proceso de DS. Por ejemplo, en [4] Jansen y Bosch introducen el concepto 'evaporación de conocimiento' para referirse al hecho por el cual prácticamente todo el conocimiento e información acerca de las decisiones de diseño en las que está basada la arquitectura, está implícitamente embebido en la arquitectura misma, y carece de una representación de primera clase. Ellos afirman que, una vez tomada la decisión de diseño, no queda huella en la Arquitectura de Software que vincule al efecto resultante en la misma con la decisión de diseño que ocasionó dicho efecto.

Buena parte de estos trabajos se enfocan en la identificación y representación de este tipo de conocimiento y otros se enfocan además en su utilización para soportar al proceso de DS.

En la primera categoría se destacan los aportes de Jansen y Bosch quienes en [4], además de introducir el concepto de "evaporación de conocimiento", definen Arquitectura de Software como una composición de decisiones arquitecturales de diseño, a las que definen como la descripción de adiciones, substracciones y modificaciones arquitecturales. En este trabajo, coincidimos en el diagnóstico y en la definición de Arquitectura de Software. Inclusive en la vinculación de las decisiones en términos de acciones sobre la arquitectura. Sin embargo, lo hacemos utilizando abstracciones en términos de un mecanismo estándar de representación de este conocimiento y en el marco del enfoque MDA.

Otro trabajo destacado en esta categoría es el que presentan Ton That et al. en [5]. Allí los autores abordan el problema desde la misma perspectiva que Jansen y Bosch [4], pero se enfocan además en mantener el vínculo entre decisiones arquitecturales abstraídas por patrones y los efectos sobre las arquitecturas de software resultantes. En este caso, coincidimos con la utilización de patrones como forma estándar de representación de este tipo de conocimiento. También coincidimos con la necesidad de persistir el vínculo, sin embargo, en nuestro trabajo vamos más allá con esta persistencia. Proponemos tipificar el vínculo como una asociación clase-instancia y la representación de primera clase tanto del patrón como de la Arquitectura de Software. Esto hace posible garantizar la consistencia de las ocurrencias de la asociación y su tratamiento para operar sobre las mismas.

En la segunda categoría, se destaca el trabajo realizado por Zimmermann et al. quienes en [6] toman las definiciones de Jansen y Bosch [4] y proponen su representación mediante el uso de plantillas. Además, proponen utilizar este conocimiento para soportar al proceso de DS mediante un metamodelo común a la representación y el reforzamiento de decisiones.

A continuación, repasamos brevemente las propuestas de definición de DS y de extensión del modelo de transformaciones MDA presentadas en [2] y, a continuación, presentamos la línea de investigación que proponemos en el presente trabajo.

2. Definición de Diseño de Software

En esta sección presentamos la definición de DS resultante de trabajos desarrollados y presentados en [2]. Se apoya en las propuestas de Wand y Ralph [7] y de Frank Buschmann et al. [8], representativas del universo disponible y consistentes con la definición del término Diseño incluida en el glosario publicado en 2010 por el comité de estándares de IEEE [9]. Además, extiende estas definiciones con el enfoque de Patrones de Diseño de Software, a fin de considerar conocimiento potencialmente reutilizable por los mecanismos de toma de decisiones del proceso. Esta extensión, abstrae en la definición la esencia de la actividad de síntesis y evaluación de alternativas para la toma de decisiones (anticipa decisiones y acota el universo de evaluación de otras) en un esquema genérico que permite su concretización para resolver un problema de diseño.

Como se observa en la *Fig. 1. Modelo conceptual de diseño de software extendido*, el modelo original [11], especializado con conocimiento del dominio de DS, aportado por Buschmann et al. en [8], ha sido extendido adicionando la perspectiva de Patrones de Diseño de Software. El modelo idealmente supone que cada parte de la Arquitectura de Software de una aplicación, puede ser obtenida como una instancia de un patrón. Si bien no es ésta la realidad del estado actual, ni hay certeza de que lo vaya a ser en un estado futuro, sí hay importante evidencia de su efectividad en un buen número de casos. El modelo también distingue el fuerte grado de dependencia (ser instancia de) de la arquitectura respecto de Patrones de Diseño de Software en comparación con el resto de las dependencias.

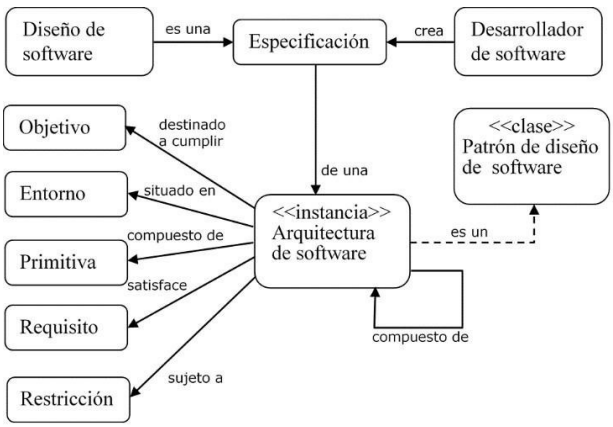

Fig. 1. Modelo conceptual de diseño de software extendido

Otro aspecto a considerar se refiere a la asociación que representa la capacidad de composición entre instancias de la clase Arquitectura de Software. Un aspecto notable es la coincidencia de esta capacidad con la capacidad equivalente entre patrones.

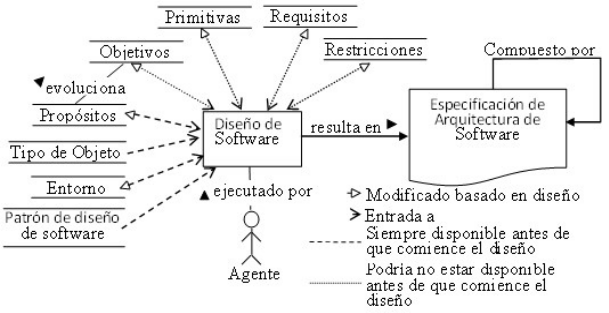

Fig. 2. Modelo conceptual del nivel contextual de diseño de software extendido con patrones

La *Fig. 2. Modelo conceptual del nivel contextual de diseño de software extendido con patrones*, muestra el nivel contextual del proceso de DS extendido con la perspectiva de Patrones de Diseño de Software. Esta extensión se refleja en la adición de una nueva entrada, Patrones de Diseño de Software, y en la estructuración del resultado mediante la asociación 'compuesto por' que permite la composición entre Arquitecturas de Software inducida por el mismo tipo de composición en Patrones de Diseño de Software. La influencia del patrón de diseño en la lógica del proceso de diseño está dada por la regla del patrón, que abstrae y simplifica la actividad de síntesis y evaluación de alternativas de la toma de decisiones.

3. El Modelo extendido

El enfoque MDA completa el modelo del PDS integrando artefactos y actividades de desarrollo mediante transformaciones. Éstas son definidas entre metamodelos para ser aplicadas a la transformación de uno o más modelos de entrada en otro de salida sin importar los niveles de abstracción definidos para la arquitectura del PDS. Una transformación puede ser utilizada para producir una representación a partir de otra, o para cruzar niveles capas arquitecturales.

La *Fig. 3. Ejemplo de Transformación Parametrizada* presenta un ejemplo de un patrón de transformación, tomado de la guía "MDA Guide Version 2.0" [10].

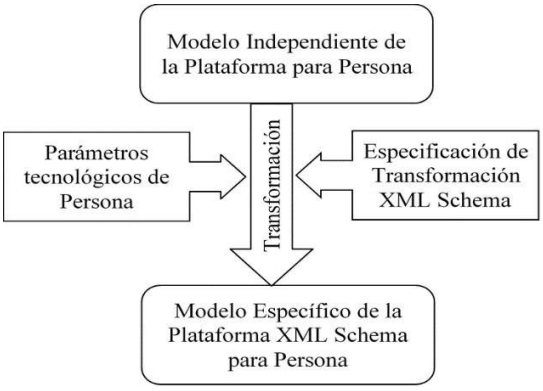

Fig. 3. Ejemplo de TransformaciónParametrizada

3.1. Modelo extendido de transformación

La *Fig. 3. Ejemplo de Transformación Parametrizada* ilustra el concepto de transformación para un hipotético caso de transformación. El modelo extendido propuesto en **[2]** presenta un modelo de transformación cuyo origen y destino se encuentran en el mismo, o diferente, nivel del modelo arquitectural del PDS. Permite obtener otra representación delmismo modelo, con transformaciones dirigidas con reglas contenidas en la especificación de un Patrón de Diseño de Software.

La siguiente *Fig. 4. Modeloextendido de transformación* presenta el modelo presentado en [2]. Este modelo extiende el modelo presentado en la *Fig. 3. Ejemplo de Transformación Parametrizada*, con el conocimiento de la *Definición de Diseño de Software* discutida. Este conocimiento, en una instancia de la transformación, es inyectado como una representación de una instancia del metamodelo de un Patrón de Diseño de Software e interpretado por la transformación que lo utiliza como guía para construir la salida.

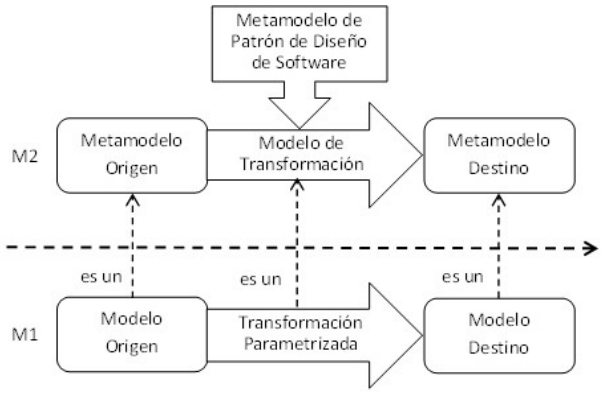

Fig. 4. Modelo extendido detransformación

El modelo asimila en la lógica de transformación la capacidad de incorporar decisiones anticipadas por el patrón y orientar la toma de otras decisiones en base a las restricciones, también aportadas por el patrón, sobre el universo potencial de evaluación de éstas.

4. Línea de Investigación y Desarrollo

Como describimos en la *Introducción*, el enfoque MDA, contribuido por OMG, proporciona un modelo del PDS que actúa como una meta-arquitectura, la cual permite integrar nuevo conocimiento mediante la contribución de metamodelos en el marco del estándar. En esa sección, también mencionamos que la dificultad para que este importante avance se materialice en la calidad de instancias del PDS generadas a partir del modelo y en los productos de software resultantes de la ejecución de las mismas, radica en el nivel de abstracción del modelo que no incluye abstracciones lo suficientemente cercanas a los dominios tecnológicos y de aplicación en que se desempeñan estas instancias. En consecuencia, y derivado de resultados producidos en trabajos previos de investigación [3] [2], presentamos aquí una línea de investigación que propone el desarrollo de un metámodelo de Patrones de Diseño de Software y de su asimilación en el modelo de transformación MDA extendido, propuesto en [2]. Para este objetivo, la línea de investigación propone:

- Definición y especificación del metamodelo para Patrones de Diseño de Software en términos del estándar MDA. Este trabajo involucra la especificación, como especializaciones de las construcciones aportadas por el enfoque MDA, de meta-definiciones para soportar la especificación de roles, comportamientos, asociaciones y reglas que definen a los Patrones de Diseño de Software.
- La extensión del modelo de transformaciones MDA, definido originalmente de un metamodelo origen en otro destino, como una transformación de un metamodelo origen más un metamodelo de Patrones de Diseño de Software en otro destino.

5. Resultados y Objetivos

Esta línea de investigación, se enmarca en el área de trabajo iniciada con las primeras líneas de investigación y trabajos desarrollados a partir del año 2012 [11] [12]. Desde entonces y hasta la actualidad, se han desarrollado diferentes trabajos que han

derivado en publicaciones de resultados [12] [2] y en trabajos de tesis en carreras de la UNSL.

Entre los trabajos en curso y planificados para el corriente año podemos destacar:

1) Durante el año 2016 se realizaron diferentes experiencias con:
   a. Especificaciones de metamodelos
   b. Especificaciones de Esquemas XML para representaciones XMI de instancias de metamodelos
   c. Desarrollo de módulos reusables experimentales para transformaciones hipotéticas con capacidad para interpretar y procesar conocimiento contenido en representaciones XMI de instancias de modelos de patrones.
2) Actualmente se ha finalizado una tesis de la Maestría en Ingeniería de Software de la UNSL, para la cual se ha solicitado mesa, que presenta una propuesta integral de una meta- arquitectura que sintetiza resultados hasta aquí obtenidos y publicados en otros trabajos.
3) Para el año 2017 está planificado el inicio de trabajos de tesis en las carreras de Licenciatura en Ciencias de la Computación y de Ingeniería de Sistemas que se ocuparán del desarrollo de mecanismos de instanciación Arquitecturas de Software a partir de patrones basados en técnicas Orientadas a Objetos en un caso y en técnicas de XML Schemas en otro.
4) En el corriente año, están planeados los trabajos, y la publicación de sus resultados, propuestos aquí, que incorporen resultados de las experiencias descriptas en 1).

6. Formación de Recursos Humanos

En el área de la temática planteada, se desarrollan diferentes actividades que van desde la inclusión de trabajos prácticos, laboratorios y ejercicios de investigación, en materias de las carreras afines hasta el desarrollo de tesis y trabajos finales, algunos finalizados y otros en curso, en las mismas. También, en trabajos de extensión, consistentes en desarrollo de aplicaciones para el medio, se desarrollan experiencias manuales asistiendo al desarrollo con refinamiento de modelos con transformaciones dirigidas por patrones, cuyos resultados son retroalimentados a los trabajos de investigación.

7. Bibliografía

[1] Object Management Group, OMG's MetaObject Facility, http://www.omg.org/mof/; 2016.

[2] L. Roqué Fourcade et al., Software Design: Towards a Meta-architecture to Support Decision Making in the Definition of Software Architecture, *SEDE 2016*, Denver, USA, 2016.

[3] L. Roqué Fourcade et al., Extendiendo la meta-arquitectura aportada por el enfoque MDA con conocimiento del dominio, WICC 2016, Entre Ríos, Argentina, 2016.

[4] A. Jansen y J. Bosch, Software Architecture as a Set of Architectural Design Decisions, *WICSA 2005*, Pittsburgh, PA, USA, 2005.

[5] Ton That et al., Preserving architectural decisions through architectural patterns, *Automated Software Engineering*, Singapore, 2016.

[6] O. Zimmermann et al., Reusable Architectural Decision Models for Enterprise Application Development, *Soft. Architectures, Components and Applications*, Medford, Springer-Verlag Berlin Heidelberg, 2007, pp. 15-32.

[7] P. Ralph y Y. Wand, A proposal for a formal definition of the design concept, *LNBIP* 2009.

[8] F. Buschmann et al., Pattern-Oriented Software Architecture - A System of Patterns, Chichester, England, John Wiley & Sons, 2001.

[9] IEEE Standards Association (IEEE-SA) Standards Board, ISO/IEC/IEEE 24765, Systems and software engineering — Vocabulary, IEEE Computer Society Press, 2010.

[10] Object Management Group, MDA Guide revision 2.0, http://www.omg.org/cgi-bin/doc?ormsc/14-06-01. 2014.

[11] L. Roqué Fourcace y L. Arakaki, Derivando el diseño a partir de especificaciones de requisitos basadas en Casos de Uso, *WICC 2012*, Posadas, Misiones, Argentina, 2012.

[12] L. Roqué Fourcade y L. Arakaki, Derivando el Diseño a Partir de Especificaciones de Requisitos Basadas en Casos de Uso, *ASSE 2012*, La Plata, Argentina, 2012.