Formalización de Descripciones de Casos de Uso a través de Metamodelos (ForCUPIDo)

Laura Aballay, María Inés Lund, Emilio Ormeño, Sandra Oviedo, Cecilia Marcuzzi, Sabrina Cruz Introini, Viviana Alferillo

Instituto de Informática – Fac. Cs. Exactas F. y Naturales – Universidad Nacional de San Juan

Ignacio de la Roza y Meglioli. Complejo CUIM. UNSJ

Tel 0264-4265101

laballay, mlund, eormeno {@iinfo.unsj.edu.ar}

Resumen

El Desarrollo Dirigido por Modelos (MDD) plantea la transformación formal y sistemática de modelos abstractos en modelos específicos a una tecnología, hasta llegar a modelos de implementación que producen código fuente. Utilizando metamodelos que abstraen los elementos de un dominio, un diseñador puede generar diagramas para definir el espacio del problema y su solución. En proyectos anteriores se desarrollaron dos metamodelos para dominios específicos: una plantilla denominada CUPIDo que impone restricciones en la documentación de casos de uso, y una serie de herramientas para el procesamiento de lenguajes específicos del dominio. Este proyecto, bajo el MDD, propone la explotación de ambos metamodelos para producir prototipos de sistemas de software en forma semi-automática.

Contexto

El proyecto, aprobado por evaluación externa y financiado por la UNSJ -Cod E883-, se encuentra inserto en el marco de las líneas de investigación del Gabinete de Ingeniería de Software del Instituto de Informática. Además se encuentra vinculado a cátedras de las carreras de Licenciatura en Ciencias de la Computación y Licenciatura en Sistemas de Información, de la FCEFN de la Universidad Nacional de San Juan.

Introducción

en la Ingeniería de Software. A fin de disminuir el esfuerzo y aumentar la calidad del software, visto como el producto resultante, sus métodos y técnicas se encuentran en constante evolución, afectando directa e indirectamente los ciclos tradicionales de desarrollo de software.

Se han hecho muchos esfuerzos para poner más rigor, previsibilidad, y eficacia en el desarrollo del software, algunos de estos involucran la captura de información referida a requisitos, arquitectura, y aplicación de sistemas de software. Otros hacen hincapié en la reutilización de código y en el diseño de herramientas de desarrollo y testeo del software.

En todo este escenario desde hace tiempo está creciendo la aceptación, en ámbitos de la industria del software, del Desarrollo Dirigido por Modelos (MDD). Esta incipiente área en el campo de la Ingeniería de Software plantea una nueva forma de entender el desarrollo y mantenimiento de sistemas de software con el uso de modelos como principales artefactos del proceso de desarrollo. En MDD los modelos son utilizados para dirigir las tareas de comprensión, diseño, construcción, pruebas, despliegue, operación, administración, mantenimiento y modificación de los sistemas. Esta metodología facilita además la generación automática de gran cantidad de código que, de otra manera, necesitaría ser generado manualmente.

Mientras el uso de modelos en el diseño y testeo no es una novedad, MDD es innovador ofreciendo tecnologías cohesivas y comprensivas para el uso de modelos de diverso nivel de abstracción y, sobre todo, metamodelos que

metamodelos pueden ser gráficos, respondiendo a estándares tales como UML, o textuales, lo cual lo constituyen en lo que se denomina Lenguaje Específico del Dominio (DSL).[1][2]

En este sentido, en los proyectos anteriores CuProSoft: Modelo de casos de uso, un eje para el proceso de desarrollo de software (E822), MeROdeA: Medición de reutilización de objetos de aprendizaje (E-827) y SeMOdelA: Especificación de semántica de metadatos de objetos de aprendizaje (E-828) se desarrollaron dos metamodelos para dominios específicos: una plantilla llamada CUPIDo [3] que impone restricciones en la documentación de casos de uso, y una serie de herramientas para el procesamiento de lenguajes específicos del dominio, como CMTL (Course Modeling Technique Language) que sirve para modelar procesos de enseñanza-aprendizaje. Este proyecto, a la luz de la técnica de MDD, propone la explotación de ambos metamodelos para producir prototipos de sistemas de software en forma semi-automática.

El Desarrollo Dirigido por Modelos [4] es un nuevo paradigma que promete mejorar la construcción de software basándose en procesos dirigidos por modelos y apoyado en herramientas. Los modelos se generan desde los más abstractos hasta los más concretos a través de transformaciones y/o refinamientos hasta llegar al código fuente como última transformación. Los elementos clave que comprenden este enfoque se pueden resumir en:

- Abstracción: El proceso plantea el desarrollo o la utilización de lenguajes amigables de alto nivel de abstracción, que se ubican más cerca del usuario final y por ende, del dominio del problema y lejos de tecnologías puntuales. Los modelos o diseños específicos que surgen de esos lenguajes (metamodelos) constituyen la base a partir de la cual, a través de una serie de transformaciones y refinamientos, surgen otros modelos más específicos y de menor abstracción, hasta llegar al código fuente como el último de los artefactos del proceso de desarrollo.

- Automatización: En todo el proceso, se hace hincapié en la necesidad de contar con procesos automáticos y herramientas computacionales que realicen todas las transformaciones desde los modelos más como el modelo más concreto. El proceso puede finalizar con la generación de programas ejecutables o incluso del testing y el despliegue en plataformas tecnológicas específicas.

- Estándares: En MDD es fundamental que todos los modelos y herramientas de transformación respeten estándares a fin de que puedan interoperar entre diversos fabricantes.

MDD identifica distintos tipos de modelo con diferente grado de abstracción [5]:

1. CIM (Computational Independent Model). Son modelos de muy alto nivel de abstracción que surgen a partir de lenguajes específicos del dominio (DSL) o de Metamodelos tales como BPMN. Este tipo de modelo tiene la ventaja de estar muy cerca del dominio del problema.

2. PIM (Plataform Independent Model). Los modelos anteriores se transforman en modelos que siguen patrones de diseño o estilos arquitectónicos tales como capas, fachada, etc. El objetivo de este tipo de modelos es definir y controlar, a través de patrones, atributos de calidad tales como extensibilidad, performance, seguridad, escalabilidad, etc.

3. PSM (Plataform Specific Model). Los modelos anteriores se transforman en modelos específicos a una plataforma o tecnología, tales como J2EE [6], .NET [7]. O frameworks que implementan patrones tale como, Enterprise Java Beans (EJB) [8], Hibernate [9], Spring [10], etc.

4. IM (Implementation Model). Estos modelos representan código fuente en sí mismo, que surgen a partir de los modelos específicos de una plataforma.

En el desarrollo tradicional las transformaciones son realizadas por humanos. Los códigos generados por algunas herramientas son sólo esqueletos de código que se deben completar manualmente. Esto lleva a que usualmente el diseño se desfase del código fuente, generando problemas de trazabilidad y de mantenimiento. Por el contrario, MDD plantea la transformación en forma automática desde los modelos de mayor

MDD consiste en la evolución de técnicas tradicionales de la Ingeniería de Software pero con el agregado de las transformaciones automáticas. Edsger Dijkstra en su libro "A Discipline of Programming" [11] define el término de refinamiento como una transformación semánticamente correcta de un diseño abstracto en otro más detallado.

Lo que diferencia a MDD respecto de las especificaciones formales utilizando cálculos de refinamiento [1][2], es que las transformaciones pueden convertir un modelo en cualquier otro lenguaje, por ejemplo, un modelo en UML puede convertirse en una base de datos específica.

El Modelo de Casos de Uso constituye uno de los modelos más utilizados por la industria del Software debido a su simplicidad para definir y describir la funcionalidad de un sistema desde el punto de vista de sus usuarios. En general, un modelo de casos de uso está compuesto por los actores o usuarios y la funcionalidad que el sistema les proveerá. Además, tal como dice Jacobson en [12] "El modelo de casos de uso permite que los desarrolladores del software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. Proporciona la entrada fundamental para el análisis, el diseño y las pruebas."

Desde otra perspectiva un tanto informal, se puede decir que un caso de uso describe la forma en que un sistema es empleado por los actores (usuarios del sistema) para alcanzar sus objetivos. Tiene una notación gráfica y está acompañado de una descripción de lo que hace [13]. Generalmente esta descripción se expresa en plantillas que permiten guiar, portar o construir un esquema predefinido y reflejar los elementos comunes de los Casos de Uso [14].

Las plantillas de casos de uso son documentos estructurados que facilitan el proceso de documentación de un proyecto de software, por lo que deberían responder a estándares internacionales que normen su formato y contenido. Debido a la carencia de un estándar para documentar casos de uso, en el proyecto CuProSoft se desarrolló una plantilla para documentar y describir casos de uso en etapas tempranas del ciclo de desarrollo, denominada CUPIDo (Casos de Uso, Plantilla Integradora para [illegible]

uso, de autores reconocidos a nivel nacional e internacional ([15] [16]), se adapta a las necesidades y ámbitos de desarrollo de software local y/o regional, en el contexto de los sistemas administrativos, donde la formalización y documentación de sistemas es escasa o nula y cuando existe es netamente informal.

Esta propuesta tiene como especial característica que plantea un principio de formalización en las descripciones de los casos de uso, especialmente en las secuencias y procesos definidos, proponiendo conceptualizaciones del dominio y una gramática, elementos totalmente compatibles con frameworks, vistas y metamodelos que pueden ser formalizados completamente.

Líneas de investigación y desarrollo

El propósito del proyecto es Generar Metamodelos y/o Lenguajes Específicos del Domino (DSLs) para formalizar descripciones de secuencias y procesos en documentos de Casos de Uso, especificados a través de la plantilla CUPIDo, a fin obtener Modelos Computacionalmente Independientes (CIMs) que sirvan para prototipar en forma semi-automática Sistemas de Software, dentro de un proceso de Desarrollo de Software Dirigido por Modelos (MDD).

Para cumplir con lo propuesto, se dividen tres líneas específicas:

- Conceptualización de elementos comunes en descripciones de casos de uso de un dominio específico.
- Desarrollo de un lenguaje formal específico al dominio de casos de uso seleccionado.
- Implementación de transformadores en herramientas de software que soportan MDD y metamodelado.

Resultados y Objetivos

En el transcurso del primer año del proyecto se realizaron varias capacitaciones a miembros del mismo, relacionadas a las temáticas involucradas:

- Curso Despegando con GWT dictado por la empresa Logikas, 20 de Junio al 8 de Agosto de 2011, abarcando los módulos: Toma de contacto, Interacción con el servidor, Creación de Interfaz de usuario, Interfaces de usuario declarativas, Internacionalización, repositorio de recursos y testing: L Aballay, S Cruz Introini, M I Lund

- Curso de Posgrado HCI (Interacción Humano Computador): L Aballay, S Oviedo, M I Lund

- Curso Programación Java- Proyecto "Cursos Interuniversitarios para el intercambio del conocimiento en el Mercosur". Del 7 al 10 de Noviembre por docentes de la Universidad de Brasil-UNISINOS-, 20hs: S Cruz Introini.

- Curso de Programación Java Junior, con beca Control-F otorgadas por Casetic. 80hs: S Cruz Introini

- Curso "Búsqueda Inteligente de Información en la Web" en la XV Escuela de Informática, y al XVII CACIC, La Plata del 10 al 14 de Octubre de 2011: S Cruz Introini

Se realizaron las siguientes publicaciones en Congresos y Revistas del área:

- Cacic 2011: "Validación de Usabilidad de una plantilla para documentar Casos de Uso - Estudio Exploratorio". M I Lund, L Aballay, E Torres, M Herrera, E Ormeño

- IHC+CLIHC 2011: “Modelo Colaborativo y Distribuido para Enseñanza de la Usabilidad”. L Aballay, C Collazos y M Herrera

- Revista RUTIC 2012 vol.1 Nº1 (de la rama estudiantil de IEEE). “Validación de un modelo de Enseñanza Colaborativo y Distribuido”. Rama Estudiantil IEEE Universidad del Cauca. Colombia. Autores: L. Aballay, C Collazos, M Herrera

- ENIDI2011: “Aplicando Logística Inversa en Manufactura”. S Oviedo, R Forradellas

Dictado Cursos:

- De Posgrado “Dinámica de Sistemas”. S Oviedo. Febrero 2012.

- De Posgrado “Desarrollo de aplicaciones con Spring Roo” en la Escuela de Ingeniería de la Universidad ORT Uruguay: E Ormeño

- “Taller de Elicitación de Requisitos en Entornos Colocalizados". S Zapata, M Reus, L Aballay

Objetivos cumplidos:

1. Se definió el dominio de estudio y el conjunto de requisitos relativos a Sistemas de Información vinculados al área administrativa.

2. Se realizó la búsqueda y recopilación de múltiples sistemas documentados por medio de descripciones de casos de uso para abstraer características, secuencias, procesos, terminología y conceptos comunes.

3. Se elaboró la primera versión del lenguaje formal para el Flujo de eventos de casos de uso.

4. Se estudiaron los servicios RESTful, herramientas como Spring y SpringRoo y se generó una primera versión de un metamodelo de UML para modelar servicios RESTful, basados en controladores y artefactos generados por SpringRoo.

5. Se realizó el modelado para la automatización de la plantilla CUPIDo.

Objetivos a Futuro

1. Someter la primera versión del lenguaje a referato nacional/internacional.

2. Adaptación de la primera versión del lenguaje a la plantilla CUPIDo.

3. Desarrollo de Software para automatizar la plantilla CUPIDo con las restricciones de formalización impuestas por el lenguaje.

4. Implementación, en lenguajes estándares, de transformadores para la generación de modelos de menor nivel de abstracción tales como diagramas de secuencias, a partir del flujo de eventos de los casos de uso.

5. Someter la implementación de los métodos y estrategias de transformación a referato nacional/internacional.

6. Implementación de nuevos transformadores específicos a plataformas tales como J2EE o .NET.

7. Difusión de los resultados, experiencias y recomendaciones en publicaciones con referato.

Los resultados parciales y finales del proyecto tienen transferencia directa a los alumnos de grado, especialmente los alumnos cursantes de la cátedra Diseño de Software, de las carreras Licenciatura en Ciencias de la Computación (LCC) y Licenciatura en Sistemas de Información (LSI), en donde utilizan como artefacto de especificación de requisitos, las plantillas de Casos de Uso. Se planea dictar cursos o talleres para que aprendan nuevas técnicas para el diseño, como las brindadas por MDD, y los resultados obtenidos en el proyecto.

Formación de Recursos Humanos

- Obtención del título de Licenciatura en Ciencias de la Información: L Aballay
- Obtención de Beca de Investigación y Creación categoría 'Estudiantes Avanzados': C Marcuzzi con el tema propuesto: "Identificación y formalización de patrones de usabilidad para el diseño de interfaces"
- Tesis de posgrado de maestría presentada para evaluación y a la espera de definición de fecha para la defensa. M I Lund.
- Tres alumnas de la carrera Licenciatura en Ciencias de la Información, adscriptas al proyecto, trabajan para realización de su tesis de grado.

Referencias

[1] R.-J. Back y J. Wright, Refinement Calculus: A Systematic Introduction (Texts in Computer Science), 1a ed. Springer, 1998.

[2] L. M. G. Feijs y H. B. M. Jonkers, Formal Specification and Design (Cambridge Tracts in Theoretical Computer Science), 1.a ed. Cambridge University Press, 2005.

[3] M. I. Lund, C. Ferrarini, L. Aballay, y E. Meni, CUPIDo - Plantilla para Documentar Casos de Uso, presented at the V Congreso de Tecnología en Educación y Educación en Argentina, 2010.

[4] D. S. Frankel, J. Parodi, y R. Soley, The MDA Journal: Model Driven Architecture Straight From The Masters. Meghan Kiffer Pr, 2004.

[5] C. Pons, R. Giandini, y G. Pérez, Desarrollo de Software Dirigido por Modelos. Conceptos teóricos y su aplicación práctica. EDUNLP and McGraw-Hill Education, 2010.

[6] «Java EE at a Glance». [Online]. Available: http://tinyurl.com/j2ee2010. [Accessed: 17-nov-2010].

[7] «Microsoft .NET Framework». [Online]. Available: http://www.microsoft.com/net/. [Accessed: 17-nov-2010].

[8] «Enterprise JavaBeans Technology». [Online]. Available: http://tinyurl.com/EJB3-0-2010. [Accessed: 17-nov-2010].

[9] «Hibernate - JBoss Community». [Online]. Available: http://www.hibernate.org/. [Accessed: 17-nov-2010].

[10] «SpringSource.org». [Online]. Available: http://www.springsource.org/. [Accessed: 17-nov-2010].

[11] E. W. Dijkstra, A Discipline of Programming. Prentice Hall, Inc., 1976.

[12] I. Jacobson, G. Booch, y J. Rumbaugh, «Chapter 1: The Unified Process: Use-Case Driven, Architecutre-Centric, Iterative, and Incremental», in The Unified Software Development Process, Addison-Wesley Professional, 1999, p. 5.

[13] Roger S. Pressman, «Ingeniería del Software, un enfoque Práctico, Quinta edición edición.», in El producto, México: Mc Graw Hill, 2003.

[14] C. Larman, «Chapter 6: Use-Case Model: Writing Requirements in Context», in Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, 2.a ed., Prentice Hall PTR, 2001, pp. 45–46.

[15] R. R. Hurlbut, «Managing Domain Architecture Evolution Through Adaptive Use Case and Business Rule Models», 1998.

[16] D. Kulak y E. Guiney, «Chapter 3: A Use-Case-Driven Approach to Requirements Gahtering. Requirements Specification Tools.», in Use Cases: Requirements in Context, 2.a ed., Addison-Wesley Professional, 2003, pp. 53–55.