Generación de aplicaciones basada en MDE

J.P. Marzetti, A. Cortez, D. Escuderi,C. Calabró, C. Naveda
Instituto de Informática . Facultad de Ingeniería. Universidad de Mendoza.
Departamento de Informática - Facultad de Ciencias Físico-Matemáticas y Naturales. UNSL.
jp@jpmarzetti.com.ar, cortezalberto@gmail.com, davidscuderi@hotmail.com,
cristian.calabro@gmail.com, claudia_naveda@hotmail.com

Resumen

La creciente complejidad de los sistemas informáticos exige el desarrollo de productos con niveles de calidad y productividad mayores. Esto es debido a la existencia de un contexto altamente dinámico y con acelerados cambios tecnológicos [1].

Como consecuencia de este contexto el diseño de software se enfrenta continuamente a la heterogeneidad de los sistemas. Tal es el caso de los sistemas distribuidos donde es difícil satisfacer los requisitos de escalabilidad, seguridad y eficiencia. Como contrapartida surge la idea de que si el desarrollo esta dirigido por modelos de software se obtendrán beneficios en cuanto a productividad, portabilidad, interoperabilidad y mantenimiento. El presente trabajo analiza la Ingeniería Dirigida por Modelos (MDE) como propuesta válida para cumplir con las exigencias mencionadas anteriormente. Además se estudia a los Lenguajes Específicos de Dominio (DSL) como la formalización del modelado planteado por MDE. Y avanza en el paso siguiente que es la generación de aplicaciones a partir de los modelos diseñados.

Palabras clave: MDE, Modelos, DSL, Diseño.

Contexto

El presente trabajo se realiza por el equipo de investigación del Instituto de Informática de la Facultad de Ingeniería de la Universidad de Mendoza. Los estudios se desarrollan en el marco de la línea de investigación “ Ingeniería de Software. Herramientas automáticas de código en el contexto de Ingeniería Dirigida por Modelos (MDE)”. Este proyecto también se nutre de la investigación de el proyecto 22/F822 "Ingeniería de Software: Conceptos, Métodos y Herramientas en un contexto de Ingeniería de Software en Evolución”, Departamento de Informática - Facultad de Ciencias Físico-Matemáticas y Naturales. UNSL. (Universidad Nacional de San Luis).

1. Introducción

La Ingeniería Dirigida por Modelos (MDE) es un enfoque de desarrollo de software orientado a crear modelos que se refieren mas a conceptos de un dominio en particular, que a un concepto computacional o algorítmico. En este ámbito los modelos juegan un papel preponderante ya que no sólo son simples elementos de documentación del software que se diseña, sino que son los verdaderos directores de todo el proceso de desarrollo del software. Los modelos se utilizan para definir la implementación, las transformaciones, los aspectos de los artefactos de software, los puntos de vista del sistema, y mucho más [5].

MDE tiene como objetivos:

1. Mejorar la productividad a corto plazo de los desarrolladores: Se incrementa el valor de los artefactos de software primarios en términos de cuánta funcionalidad ofrecen.

2. Mejorar la productividad a largo plazo de los desarrolladores: Se incrementa el tiempo en el cual dichos artefactos se vuelven obsoletos reduciendo la sensibilidad de los artefactos de software a: cambio de personal, requerimientos, plataformas de desarrollo, plataformas de despliegue.

Según MDE, la calidad del software puede ser comprobada y asegurada con tres técnicas diferentes: validación de modelos, comprobación de modelos y pruebas basadas en el modelo [5].
Un enfoque que busque cumplir con los objetivos de MDE debe tener más que la dimensión abstracta y concreta que generalmente se utiliza para diseñar sistemas. En estas dimensiones sólo se puede discriminar si un modelo es independiente de la plataforma con respecto a otro modelo o no. Kent define dos categorías adicionales de dimensiones que son necesarias para trabajar con MDE [11]. En la primera categoría se incluye:

- La distinción de modelos sobre la base del área a la que pertenecen: diferentes usuarios pueden tener distintos puntos de vista del sistema focalizados en un subconjunto de las funcionalidades del mismo. Por ejemplo: el módulo de ventas, el de stock, el de clientes.

- La enumeración de aspectos del sistema: ejemplos de esta dimensión podrían ser información/datos, presentación, control de concurrencia, seguridad, distribución y manejo de errores.

En la segunda categoría se puede encontrar dimensiones que están menos relacionadas con los aspectos técnicos del sistema pero corresponden a asuntos organizacionales. Las dimensiones en esta categoría incluyen: autoría, control de versiones y de configuración, ubicación (en el caso que el sistema sea desarrollado de manera distribuida) e interesados (expertos de negocio o desarrolladores). En los proyectos de desarrollo de software es necesario determinar las dimensiones importantes para cada caso particular.
Otra decisión importante es el nivel de abstracción necesario para ser utilizado en el proceso y quienes estarán involucrados en el mismo. Esto permite definir cómo serán los modelos para que todo el mundo pueda entenderlos [3].
Para concretar los modelos se utilizan metamodelos, es decir, modelos en un lenguaje específico que definen la estructura de la aplicación. Como resultado un metamodelo A define el lenguaje de un modelo B. Y en consecuencia, los elementos del modelo B son instancias lingüísticas de los elementos del metamodelo A.

Lenguajes específicos de Dominio

Al hablar de MDE y Metamodelado se ha mencionado que estos conceptos se extienden más allá de la dimensión concreto/abstracto. De esta manera se define un marco de trabajo de múltiples dimensiones e intersecciones, y con ello , la posibilidad de expresión a través de diferentes modelos.
Las dimensiones que se pueden encontrar en una intersección juegan un papel importante en la elección de un lenguaje de modelado para un modelo particular. Por ejemplo: un lenguaje de modelado puede estar influenciado por las áreas de interés, los

interesados, y el nivel de abstracción. En dicho caso se puede hablar del mencionado lenguaje como un Lenguaje Específico de Dominio (DSL).
Un DSL puede ser definido como un lenguaje de programación o la especificación de un lenguaje ejecutable que ofrece a través de notaciones apropiadas y abstracciones un poder de expresión focalizado y usualmente restringido a un cierto dominio de problemas.
La especificidad de dominio de un lenguaje no es una medida absoluta. Cualquier lenguaje tiene un cierto ámbito de aplicación aunque algunos pueden ser más restringidos que otros [6].

1. Líneas de investigación y desarrollo

Se plantea el estudio de MDE como una alternativa favorable y viable para ejecutar proyectos de software tanto a corto como a largo plazo. Este estudio ha llevado a profundizar en los elementos que permiten implementar MDE para el desarrollo de software de manera más automatizada.
La investigación se ha centrado en el estudio de los lenguajes específicos de dominio como herramienta de expresión de modelos y todo lo referente a las transformaciones de estos en aplicaciones. Se ha comenzado el desarrollo de UMGénesis, un lenguaje específico de dominio que permita modelar aplicaciones basadas en acceso a datos. En las bases de su diseño se buscar permitir modelar rápidamente la mayor parte de la aplicación, incluyendo una vista estática del modelo de datos y una vista dinámica del trabajo sobre ellos. Permite definir ciertas reglas de negocio, validación de datos e integridad de los mismos.
El modelo dinámico utiliza una definición interconectada de sus unidades de trabajo llamadas ***usos,***que permite definir el flujo de los mismos en la interacción con el usuario. Para los casos de funcionalidad no contemplada dentro del modelo, UMGénesis permite definir funcionalidades extra que serán implementadas durante la fase de codificación manual.

2. Resultados y Objetivos

Después de una ardua investigación se concluyó en la elección de un generador de aplicaciones Java para materializar el lenguaje UMGenesis. Todo el desarrollo se está realizando con los frameworks que el entorno de desarrollo Eclipse ofrece para modelado. (Eclipse, Eclipse Modelling Framework).
Entre las herramientas utilizadas están: XText (utilizado para diseñar el DSL, su editor, compilador e instanciador del modelo), XPand y XTend (utilizado para realizar transformaciones de modelos y generar los códigos fuentes) y OCL (utilizado para realizar las validaciones de los modelos) (Eclipse, XText Documentation).
El trabajo de desarrollo, prueba y revisión es facilitado enormemente por el uso del framework de Xtext. Si se considera que éste genera todo el código necesario para el editor, el analizador y el compilador, el hacer todo este proceso de forma manual sería muy costoso [1].
Xtext provee una forma de trabajo y la arquitectura e implementaciones necesarias para desarrollar un producto de calidad que se integra perfectamente con el entorno Eclipse permitiendo rápidamente desarrollar un DSL.
El desarrollo más importante de UMGénesis actualmente se da en el ámbito de los generadores y la arquitectura del sistema de generación de código. Su diseño es tal de manera que sea fácil y simple la implementación de nuevos generadores

permitiendo una gran flexibilidad en el proceso de generación de código.

Los generadores se implementan como fragmentos del plug-in de generación de código. Esto otorga una gran facilidad de uso e instalación de los generadores que son detectados automáticamente por el sistema de generación de código y habilitados para su uso.

En este momento se ha comenzado a implementar las transformaciones del modelo necesarias para plasmar el generador de aplicaciones utilizando *Java Server Faces, Hibernate y Rich Faces.*

3. Formación de recursos humanos

En el marco de este proyecto se está desarrollando una tesis de grado. Como corolario de la misma se construyó un proyecto de aplicación en una materia optativa denominada Ingeniería de Software Aplicada y la materia fue dictada en la Facultad de Ingeniería de la Universidad de Mendoza. Adicionalmente existe en curso otra tesis de grado que continua con la investigación planteada pero anexa al modelado la temática de lenguajes gráficos.

4. Referencias

[1] Andino Luciano Omar y Ruiz Germán Estéban (2009). Análisis y uso de los frameworks de Eclipse para la definición de DSLs. Universidad Nacional de La Plata.

[2] Atkinson Colin y Kühne Thomas (2003). Model-Driven Development: A Metamodeling Foundation. IEEE Software.

[3] Atkinson Colin y Kühne Thomas (2001). Processes and products in a multi-level metamodeling architecture. International Journal of Software Engineering and Knowledge Engineering.

[4] Den Haan Joan (2008). http://www.theenterprisearchitect.eu: “Model Driven Engineering”

[5] Den Haan Joan (2009). http://www.theenterprisearchitect.eu: “MDE - Model Driven Engineering - reference guide”

[6] Den Haan Joan (2008). http://www.theenterprisearchitect.eu: “Combining general purpose languages and domain specific languages for Model Driven Engineering”

[7] Den Haan Joan (2008). http://www.theenterprisearchitect.eu: “DSL and MDE, necessary assets for Model-Driven approaches”

[8] Eclipse Foundation. Eclipse Modeling Framework. http://www.eclipse.com/emf/.

[9] Eclipse Foundation. Xtext Documentation.http://www.eclipse.org/Xtext/documentation/0_7_2/xtext.html

[10] HUMAN PERFORMANCE CENTER (HPC),Arquitectura de Software,(2002). https://www.spider.hpc.navy.mil/index.cfm?RID=TTE_OT_1000025

[11] Kent Stuart (2002). “Model Driven Engineering”. Springer.