ESTRATEGIAS PARA EL TRATAMIENTO DE ANTIPATRONES EN LOS MODELOS DE SOFTWARE.

A.Cortez, C. Naveda

Instituto de Investigaciones – Facultad de Ciencias Administrativas y Sociales. UDA.
Instituto de Informática. Facultad de Ingeniería. Universidad de Mendoza.
Departamento de Sistemas - Universidad Tecnológica Nacional - FRM
Mendoza-Argentina

cortezalberto@gmail.com, claudia_naveda@hotmail.com

Resumen

Como definición práctica se puede decir que los antipatrones son ejemplos bien documentados de malas soluciones para problemas. Se estudian a fin de poderlos evitar en el futuro. En particular, para el ámbito de la ingeniería de software, los antipatrones describen prácticas repetidas en las aplicaciones que conducen a soluciones funestas para los proyectos. Por lo cual, el objetivo de esta investigación es generar estrategias para manejar los problemas que creados a partir de la existencia de antipatrones en un modelo y/o aplicación de software.

Palabras clave: antipatrones, estrategia, tratamiento.

Contexto

El proyecto de investigación “Creación de una herramienta generadora de código automático que valide y verifique antipatrones” se desarrolla en el marco de la línea de investigación de “Ingeniería de de Software. Técnicas avanzadas en MDA”. El mismo es avalado por Instituto de Investigaciones FCSA – UDA (Universidad del Aconcagua). La idea surge a partir de la necesidad de mejoras en el desarrollo de software planteada en distintos trabajos de tesis. Se observó la insuficiencia de trabajos al respecto como un tema preocupante para las empresas y se decidió plantear la situación para su escudriñamiento.

Este proyecto se nutre de conocimientos de la línea de investigación “Ingeniería de Software. Herramientas automáticas de código en el contexto de Ingeniería Dirigida por Modelos (MDE)”, del Instituto de Informática de la Facultad de Ingeniería de la Universidad de Mendoza.

Además cuenta con el apoyo del Departamento de Sistemas de la Universidad Tecnológica Nacional – FRM – Mendoza, donde se dicta la cátedra optativa “Arquitectura de Software dirigida por Modelos”.

1. Introducción

Los antipatrones (*antipatterns*) son descripciones de situaciones, o soluciones, recurrentes que producen consecuencias negativas. Un antipatrón puede ser el resultado de una decisión equivocada sobre cómo resolver un determinado problema, o bien, la aplicación correcta de un patrón de diseño en el contexto equivocado.

Al igual que los patrones de diseño, los antipatrones, proveen un vocabulario común con el fin de mejorar la comunicación en el equipo de desarrollo, de manera tal de poder identificar problemas y discutir soluciones.

En la elaboración de un sistema, intervienen al menos, diversos actores: arquitectos de software, administradores de proyecto y desarrolladores. Para cada uno de ellos, existen antipatrones que describen comportamientos y soluciones incorrectas. Los antipatrones (una vez conocidos) constituyen descripciones de problemas recurrentes en la construcción de software y proporcionan un vocabulario común para identificar problemas y discutir posibles soluciones. Como corolario se obtienen pasos para la reingeniería, y reorganización estructural de un sistema.

El modelado específico de dominio DSM (Domain-Specific Modeling en Ingles) es una metodología de la ingeniería de software cuyo propósito es crear modelos para un dominio, utilizando un lenguaje orientado y especializado en ese dominio. El modelado específico también incluye la idea de generación automática de código y la creación de código ejecutable directamente desde los modelos. De esta manera, las aplicaciones finales serán luego generadas a partir de estas especificaciones en alto nivel. Estos lenguajes se denominan DSLs (por su nombre en ingles: Domain-Specific Language) y permiten especificar la solución usando directamente conceptos del dominio del problema. Los DSLs generan modelos específicos de dominio. Dichos modelos son el resultado de especificar un modelo en un lenguaje específico de dominio. Esta técnica eleva el nivel de abstracción más allá de la programación especificando la solución directamente con los conceptos del dominio. Los productos finales son generados desde las especificaciones a este nivel de abstracción.

La validación de modelos es una técnica utilizada para evaluar modelos con respecto a un criterio semántico y sintáctico. Implica una verificación de consistencia, es decir: evaluar el modelo contra su metamodelo. Además de los metamodelos, pueden utilizarse restricciones para validar el modelo. El lenguaje más utilizado para este propósito es un estándar de OMG llamado OCL (Object Constraint Language) que consiste en un lenguaje formal para describir expresiones sobre los modelos.

El objetivo de este proyecto es tanto

investigar la detección y eliminación de antipatrones como la verificación y validación de modelos.

2. Líneas de investigación y desarrollo

En el marco de este proyecto se investigará

a) La detección y eliminación de antipatrones en modelos de software.
b) La validación y verificación de modelos.
c) La generación de código automático.

El objeto de estudio de esta investigación implica la construcción de nuevos objetos como lo son las herramientas de generación de código. Una metodología de investigación debe tomarse como una guía, pero no como algo rígido; debe adaptarse para cada utilización de la misma. El método se va haciendo, refinando, a medida que se avanza en la resolución del problema. Así, en este caso la definición del método de trabajo no concluye hasta que se finaliza la fase de resolución y verificación. La fase de resolución mostrará cómo resolver el problema y, la fase de definición del método de investigación, se irá haciendo y refinando en aproximaciones sucesivas.

3. Resultados y Objetivos

El proyecto se encuentra en su fase inicial desde marzo de 2011 y se prevén las siguientes tareas para ser realizadas en su primer año:

a) Refinamiento e investigación documental orientada a la identificación de trabajos previos vinculados a la detección de antipatrones.

b) Identificación de las últimas técnicas utilizadas en el área de la detección de antipatrones.

a) Identificación de casos de estudio aceptados por la comunidad científica en general para ser utilizados en pruebas de concepto y validación del proyecto.

b) Identificación y delimitación de problemas vinculados a la búsqueda de patrones mediante estas técnicas.

c) Creación de ejemplos de aplicación que prueben su utilidad concreta.

d) Investigar y seleccionar herramientas para ejemplificar el código de cada patrón y elaborar las soluciones correspondientes a cada antipatrón.

e) Formular ejemplos teóricos y prácticos que contribuyan al entendimiento de los problemas de diseño.

De acuerdo a la investigación realizada sobre

herramientas para el punto d), se concluyó en el encuentro del hallazgo de algunas herramientas que describimos a continuación. El IDE Eclipse junto con los proyectos EMF y GMF se juzgó como apropiada para la ejemplificación de código de cada patrón y elaborar las soluciones correspondientes a cada patrón.

Eclipse es una herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo (IDE(Integrated Development Environment)). Y además es una plataforma de desarrollo de software abierto (open-source).

Se encontró que es posible generar un meta-modelo utilizando EMF, donde se especifiquen los antipatrones. El proyecto *Eclipse Modeling Framework* es un framework para modelado, que permite la generación automática de código para construir herramientas y otras aplicaciones a partir de modelos de datos estructurados. En EMF los modelos se especifican usando un meta metamodelo llamado Ecore. Ecore es una implementación de MOF.

Se descubrió también las posibilidades de GMF (Graphical *Modeling Framework)* con el objetivo de generar herramientas de edición gráfica. Su función es proporcionar a los desarrolladores la posibilidad de construir editores gráficos que facilite a los diseñadores el desarrollo de aplicaciones basadas en Eclipse. A su vez se investigó las posibilidades de OCL, para asignar las restricciones de uso a cada elemento. El plugin OCL permite asegurar que el modelo este correctamente formado ya que facilita la definición de restricciones sobre el modelo Ecore para luego evaluarlas.

4. Formación de recursos humanos

En el marco del proyecto “Creación de una herramienta generadora de código automático que valide y verifique antipatrones” se está desarrollando una tesis de maestría en la Universidad de San Luis. Y articulando la elaboración de tesis de grado en las carreras de grado de las universidades locales. Se está tramitando el contacto con otras universidades a nivel local e internacional para fomentar esta línea de investigación. Para este punto ya se tiene contacto con la Universidad de Mendoza y la Universidad Tecnológica Nacional (FRM) Regional Mendoza.

5. Referencias

García Carlos Diego. 2009. Tesis "Implementación de técnicas de evaluación y refinamiento para OCL 2.0 sobre múltiples lenguajes basados en MOF".

Mohamed El-Attar and James Miller.2006 Matching Antipatterns to Improve the Quality of Use Case Models.14th IEEE International Requirements Engineering Conference (RE’06).

Brown, W. J., Malveau, R. C. y Thomas J. M. *2009.Antipatterns*. www.ibm.com/developerworks/webservices/library/ws-antipatterns. Página vigente al 2009-06-07.

John Wiley & Sons.1998.
AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.

Bill Dudney, Stephen Asbury, Joseph Krozak, and Kevin Wittkopf. 2003. J2EE AntiPatterns. Ed. John Wiley & Sons, 2003.

William J. Brown. Ed. John Wiley & Sons, 1998. Antipatterns: Refactoring Software, Architectures, and Projects in Crisis.

William J. Brown. Ed. John Wiley & Sons, 2000. Antipatterns in Project Management.

Papadimoulis and D. Shay. 2004. *Not quite getting that Object-polymorphismthing...*

URL: http://thedailywtf.com/ShowPost.aspx?PostID=23711.

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John 1994. Design Patterns: Elements of Reusable software, Addison-Wesley.

Gogolla, Martin, Bohling, Jorn and Richters, Mark 2003.Validation of UML and OCL Models by Automatic Snapshot Generation". Proc. 6th Int. Conf. Unified Modeling Language (UML'2003). Springer, Berlin, pages 265-279.

MDA 2007. Model-Driven Arquitecture. Disponible en www.omg.org/mda.

MOF 2006.Meta Object Facility. Documento: formal. Disponible en www.omg.org/mof.

Martinez, Liliana 2008.Componentes MDA para patrones de diseño. Tesis de Maestría. Universidad Nacional de La Plata.