MODELO DE CASOS DE USO. SU APLICACIÓN EN UN ENTORNO DE DESARROLLO GLOBAL DE SOFTWARE

María Inés Lund[1], Susana Beatriz Chavez[2], Emilio Gustavo Ormeño[1], Adriana Martin[2], Gerardo Matturro[3]

[1]Instituto de Informática de la FCEFN de la UNSJ
[2]Departamento de Informática de la FCEFN de la UNSJ
[3]Universidad ORT Uruguay
mlund, schavez, eormeno@{iinfo.unsj.edu.ar}, adrianamartin1@gmail.com, matturro@uni.ort.edu.uy

Resumen

El uso del Modelo de casos de uso, para plasmar los requisitos funcionales de un sistema desde el punto de vista del usuario, es una práctica cada vez más común. En general siempre han sido usados en entornos de desarrollo de software co-localizados (en el mismo lugar), y hay poca documentación sobre su aplicación en entornos de desarrollo global o distribuido (en espacios diferentes).

Los casos de uso deben ser documentados, se utilizará para ello la plantilla CUPIDo, generada y validada anteriormente por el grupo de investigación.

Se pretende con esta investigación identificar las cualidades de un proceso de desarrollo de software distribuido, analizar cómo se comporta el modelo de casos de uso en este entorno, que aspectos se deben considerar y que características deberían incluirse en la plantilla CUPIDo para adaptarse a ellos, teniendo en cuenta también si se trabaja en forma sincrónica o asincrónica, en un marco de CSCW (Trabajo Colaborativo Soportado por Computadora).

Además, y bajo la premisa del desarrollo dirigido por modelos, se propondrá un sistema para documentar las plantillas CUPIDo y un lenguaje que permita generar, desde el flujo de eventos de los casos de uso, diagramas de clases o de objetos, o de secuencias, que sirvan de modelo de entrada para las siguientes etapas del desarrollo de software.

Palabras clave: modelo de casos de uso, desarrollo global de software, plantillas CUPIDo, CSCW

Contexto

Esta temática de investigación se enmarca en los proyectos: "Modelo de casos de uso. Línea Base para el proceso de desarrollo de software", presentado en la convocatoria 2013 de proyectos de la UNSJ, cuyas unidades ejecutoras son los Instituto y Departamento de Informática, y "Red Latinoamericana de Coordinación y Cooperación para unificar buenas prácticas de desarrollo de software", aprobado por la SPU del MEN, según Resolución 2340-SPU, convocatoria de Proyectos de Fortalecimiento de Redes Interuniversitarias VI, cuyas unidades ejecutoras son: UNSJ, UCauca de Colombia, UCartagena de Colombia, UTP de Panamá, UORT de Uruguay y PUC de Chile.

Introducción

La especificación de requisitos de software es el documento básico para todo proceso de desarrollo de software, se podría decir que es el contrato con el cliente sobre lo que él espera del sistema y el sistema le debe brindar. Son la línea de base para las etapas del proceso de desarrollo de software.

Siempre bajo la premisa de que el desarrollo de software es un trabajo colaborativo, las etapas del proceso de desarrollo de software pueden ejecutarse en entornos co-localizados o distribuidos. En este último caso los miembros del equipo están ubicados en varios sitios remotos, en donde la interacción entre ellos requiere el uso de tecnología para facilitar la comunicación (que es mucho menos fluida que en grupos de desarrollo co-localizados) y la coordinación, además de mediar con un número de problemas surgidos por la distancia, el tiempo y las diferencias culturales, afectando directamente los procesos de comunicación, coordinación y de control de actividades, por ende afecta la calidad del software.

Ingeniería de software

Más allá de la definición dada por el IEEE [1] en 1990, la ingeniería de software es una disciplina emergente y está sometida a una evolución continua [2]. Depende fuertemente de la tecnología y métodos actuales y deben dar respuesta a los requerimientos de la realidad del mercado local [3]. Desde su concepción, ha sido y es una disciplina ampliamente estudiada y aplicada tanto en ambientes académicos como en los productivos. Es relativamente joven y además es completamente dinámica.

Existen numerosos procesos, técnicas, metodologías, y modelos que dan fe de ello, muchos ya convertidos en estándar, todos ellos tendientes a mejorar la calidad del proceso de desarrollo de software y en consecuencia la calidad del producto final, entre ellos se encuentra el cuerpo de conocimientos básicos de ingeniería de software, conocido como SWEBOK [2].

Un proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad" [4].

MDD

El Desarrollo Dirigido por Modelos [5] es un 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.

MDD identifica distintos tipos de modelo con diferente grado de abstracción:

1. CIM (Computational Independent Model) / Text. 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. El objetivo de estos 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, .NET. O

frameworks que implementan patrones tale como, Enterprise Java Beans (EJB), Hibernate, Spring, etc.

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

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" [6] define el término de refinamiento como una transformación semánticamente correcta de un diseño abstracto en otro más detallado.

Modelo de Caso de Uso

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. "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. El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores y proporciona la entrada fundamental para el análisis, el diseño y las pruebas" [7].

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 [8]. 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 [9].

Plantillas de Casos de Uso

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 E-822 , se elaboró una plantilla para documentar y describir casos de uso en etapas tempranas del ciclo de desarrollo, denominada CUPIDo (Casos de Uso, Plantilla Integradora para Documentarlos) [10], en base a otras de autores reconocidos a nivel nacional e internacional ([11],[12],[13]). Esta plantilla fue generada con el fin de que se pueda ir refinando y generando distintas versiones, de acuerdo a las necesidades específicas de cada etapa de desarrollo de software.

En el proyecto E-883, se refinó y se obtuvo la versión 1.4 estándar y se elaboró una nueva versión, incorporando requerimientos de diseño de interfaz y fue denominada CUPIDo+Pi (por Patrones de Interacción) [14].

Desarrollo global de software

El crecimiento vertiginoso de las Tecnologías de la Información y Comunicaciones (TICs) está generando nuevas formas de trabajo y modificando diversas prácticas en la vida cotidiana de las personas. El trabajo distribuido es una consecuencia de la globalización y representa un importante y creciente escenario laboral. La globalización de los mercados impacta fuertemente en el desarrollo de software, muchos proyectos se ejecutan en escenarios geográficamente distribuidos y el desarrollo global de software está transformándose en norma dentro de la industria del software [15].

La globalización en la Ingeniería de Software, al superar las barreras del tiempo y la distancia, permite reducir el tiempo de llegada al mercado, incrementar la productividad de los equipos, mejorar la

calidad del producto y reducir los costos de desarrollo para la organización de software [15], [16]. En el Desarrollo Distribuido de Software los miembros del equipo estén ubicados en varios sitios remotos durante el ciclo de vida del software y la interacción entre los miembros requiere el uso de tecnología para facilitar la comunicación y coordinación [16]. Cuando la comunicación traspasa los límites de una nación suele llamarse Desarrollo Global de Software.

El desarrollo global de software agrega nuevas complejidades y desafíos. Si bien existe un cuerpo de conocimientos y estándares sobre el desarrollo de software, que ha crecido, afianzado y madurado con los años, la gestión del desarrollo global y distribuido de software aún está en evolución [15], restan métodos y técnicas a desarrollar y validar, para transformarla en una disciplina más madura. Se han propuesto prácticas que permiten disminuir la distancia temporal, geográfica y socio-cultural [17], reduciendo los problemas de comunicación, coordinación y control, también prácticas claves para llevar a cabo el trabajo colaborativo a distancia [18], ó sobre cómo administrar o gestionar el conocimiento global y distribuido [19], o mejores prácticas para una ingeniería de requisitos dentro del desarrollo global de software [20], entre otras.

A partir de esta perspectiva, cobra mayor importancia el área de investigación de Trabajo Cooperativo Asistido por Computador (CSCW, por su sigla en inglés Computer Supported Collaborative Work), encaminado al estudio del ser humano dentro del contexto de trabajo, así como del diseño de herramientas (groupware) que den soporte al trabajo en grupo [21].

Líneas de Investigación, Desarrollo e Innovación

Las principales líneas de investigación son:

- Identificación de características inherentes al desarrollo global de software.
- Mapeos y revisiones sistemáticas de la literatura, respecto al desarrollo de software global e ingeniería de requisitos, y respecto al desarrollo de software global y el modelo de casos de uso como técnica de especificación de requisitos.
- Elaboración de modelos de transformación (MDD).

Resultados y Objetivos

Proyecto en etapa inicial. El objetivo principal del proyecto propuesto es obtener nuevos modelos de especificación de requisitos, para que desde la definición del modelo de casos de uso (donde se determinan los requerimientos funcionales del sistema desde el punto de vista del usuario) se pueda ir incorporando, en nuevas versiones de la plantilla CUPIDo, datos relacionados a los requerimientos y necesidades de información de las distintas etapas del desarrollo de software y considerando ambientes de trabajo co-localizados y distribuidos. Todo esto con el fin último de obtener, en forma automática o semi-automática, modelos que sean útiles a las diferentes etapas del desarrollo de software colaborando en la construcción de software de calidad.

Formación de Recursos Humanos

El grupo de investigación que se integran a estas líneas de investigación son:
4 profesores investigadores UNSJ
5 profesores otras universidades (proy SPU)
6 alumnos adscriptos (2 becarios)

A la fecha se tiene:
2 tesis de grado finalizadas (2013)
5 tesis de grado en curso (2014)
1 estancia en UNSJ de 3 meses doctorando de U.Cartagena (2014).

Referencias

[1] IEEE Computer Society, «IEEE Standard Glossary of Software Engineering Terminology. Std 610.12-1990», *IEEE Comput. Soc.*, n.º 1, p. 83, 1990.

[2] P. Bourque y R. Dupuis, «Guide to the Software Engineering Body of Knowledge 2004 Version», *IEEE Press*, p. 191, 2004.

[3] E. Bareiša, E. Karčiauskas, E. Mačikėnas, y K. Motiejūnas, «Research and development of teaching software engineering processes», en *Proceedings of the CompSysTech '07*, Bulgaria, 2007, pp. 75:1-75:6.

[4] I. Jacobson, G. Booch, y J. Rumbaugh, *The unified software development process*. Reading, Mass.: Addison-Wesley, 1999.

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

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

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

[8] R. S. Pressman, «Chapter 21: Object-Oriented Analysis», en *Software Engineering: a Practitioner's Approach*, 5Rev Ed., McGraw-Hill, 2000, pp. 571-572.

[9] C. Larman, «Chapter 6: Use-Case Model: Writing Requirements in Context», en *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.

[10] M. I. Lund, C. Ferrarini, L. Aballay, y E. Meni, «CUPIDo - Plantilla para Documentar Casos de Uso», presentado en V Congreso de Tecnología en Educación y Educación en Tecnología, El Calafate, Santa Cruz, Argentina, 2010.

[11] «Web services programming tips and tricks: Documenting a use case». [En línea]. Disponible en: http://www.ibm.com/developerworks/webservices/library/ws-tip-docusecase.html. [Accedido: 18-nov-2010].

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

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

[14] L. N. Aballay, M. I. Lund, E. G. Ormeño, S. Cruz Introini, C. Marcuzzi, y G. Jofré, «Plantilla CUPIDo: automatización y avances», presentado en XV Workshop de Investigadores en Ciencias de la Computación, 2013.

[15] D. Damian y D. Moitra, «Global Software Development: How Far Have We Come?», *IEEE Softw.*, vol. 23, n.º 5, pp. 17- 19, oct. 2006.

[16] M. Jiménez, M. Piattini, y A. Vizcaíno, «Challenges and Improvements in Distributed Software Development: A Systematic Review», *Adv. Softw. Eng.*, vol. 2009, pp. 1-14, 2009.

[17] H. Holmström, B. Fitzgerald, P. J. Ågerfalk, y E. Ó. Conchúir, «Agile Practices Reduce Distance in Global Software Development», *Inf. Syst. Manag.*, vol. 23, n.º 3, pp. 7-18, jun. 2006.

[18] J. Cusick y A. Prasad, «A Practical Management and Engineering Approach to Offshore Collaboration», *IEEE Softw.*, vol. 23, pp. 20-29, sep. 2006.

[19] K. C. Desouza, Y. Awazu, y P. Baloh, «Managing Knowledge in Global Software Development Efforts: Issues and Practices», *IEEE Softw.*, vol. 23, pp. 30-37, sep. 2006.

[20] J. M. Bhat, Mayank Gupta, y S. N. Murthy, «Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing», *IEEE Softw.*, vol. 23, n.º 5, pp. 38-44, oct. 2006.

[21] J. Grudin, «Computer-supported cooperative work: history and focus», *Computer*, vol. 27, n.º 5, pp. 19-26, may 1994.