TESTING GUIADO POR EL DISEÑO. DISEÑO DIRIGIDO POR CASOS DE USO

María Inés Lund[1], Susana Beatriz Chavez[2], Diego Checcarelli[3], Viviana Alferillo[3], Adriana Martin[2], Emilio Gustavo Ormeño[1]

[1]Instituto de Informática de la FCEFN de la UNSJ
[2]Departamento de Informática de la FCEFN de la UNSJ
[3]Alumno tesista
mlund, schavez@{iinfo.unsj.edu.ar}, diegochecarelli@hotmail.com, alferillo.viviana@gmail.com, adrianamartin1@gmail.com, eormeno@iinfo.unsj.edu.ar

Resumen

El Modelo de casos de uso es un artefacto de software muy utilizado para plasmar los requisitos funcionales de un sistema, desde el punto de vista del usuario. Los casos de uso deben ser bien documentados.

Se pretende, por un lado, identificar las cualidades de un proceso de desarrollo de software distribuido, (en forma sincrónica o asincrónica), 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 [1] para adaptarse a ellos, en un marco de CSCW.

Por otro, hacer foco especialmente en los testing de software, como parte fundamental del proceso de desarrollo, y esenciales para el éxito de la producción de software de alta calidad. Los defectos y las fallas pueden deberse a diferentes causas, como a los problemas de comunicación entre el cliente y los diseñadores, a los procesos de diseño inmaduros -desde la recopilación de requerimientos hasta el diseño detallado y arquitectura- o incluso a los malentendidos o transcripciones incorrectas de los requerimientos. Se pretende identificar características a considerar en la especificación de casos de uso, para ser usados como base en el proceso de testing, en un entorno DDT.

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

Contexto

Esta presentación se enmarca en el proyecto de investigación: "Modelo de casos de uso. Línea Base para el proceso de desarrollo de software", Aprobado y financiado por Res. N° 18/14-CS-UNSJ, cuyas unidades ejecutoras son los Instituto y Departamento de Informática.

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 lo que el sistema le debe brindar. Es 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.

Los testing deberían estar presentes en todo el ciclo de vida del software, desde su diseño hasta el final de su mantenimiento, y durante toda la operación del software. Las rigurosas pruebas de software, incluyendo su documentación, permiten una reducción de la probabilidad de fallas durante la ejecución del software y contribuyen a mejorar la calidad [2].

Es primordial, elegir metodologías de desarrollo que permitan construir aplicaciones en el menor tiempo posible, que cumplan con los requisitos de verificación y de validación del software. Esto es *verificar* que el software reúna las especificaciones establecidas y *validar* que lo diseñado haga lo que el cliente quiere [3].

Ingeniería de software

Más allá de la definición dada por el IEEE [4] en 1990, la ingeniería de software es una disciplina emergente y está sometida a una evolución continua [5]. Depende fuertemente de la tecnología y métodos actuales y debe dar respuesta a los requerimientos de la realidad del mercado local [6]. 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 [5].

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" [7].

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" [8].

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 [9]. 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 [10].

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) [11], en base a otras de autores reconocidos a nivel nacional e internacional ([12],[13],[14]). 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) [15].

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 [16].

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 [16], [17]. En el Desarrollo Distribuido de Software los miembros del equipo están 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 [17]. 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 [16], 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 [18], reduciendo los problemas de comunicación, coordinación y control, también prácticas claves para llevar a cabo el trabajo colaborativo a distancia [19], ó sobre cómo administrar o gestionar el conocimiento global y distribuido [20], o mejores prácticas para una ingeniería de requisitos dentro del

desarrollo global de software [21], 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 [22].

Testing

En particular, el testing es una actividad realizada para evaluar la calidad de un producto y mejorarlo a través de la identificación de defectos y problemas [5], [2], [23] Se lo podría definir como un conjunto de actividades que tiene por objetivo identificar las fallas en un software y evaluar su nivel de calidad para obtener la satisfacción del usuario. Se trata de un conjunto de tareas con objetivos claramente definidos [24].

Los objetivos formulados para cada testeo depende de la fase del ciclo de vida del software [25]:

- Durante el diseño general o fase de diseño detallado, los tests se centrarán en encontrar el mayor número de defectos (o fracasos), en el menor tiempo posible.
- Durante la fase de aceptación de los clientes, los tests demostrarán que el software funciona correctamente para obtener la aprobación del cliente.
- Durante la fase de operación, los tests se centrarán en asegurar que los niveles de exigencia están logrados.
- Durante el mantenimiento evolutivo y correctivo, los tests tienen como objetivo garantizar la ausencia de fallas y de efectos secundarios.

El testing ya no es visto como una actividad que se inicia sólo después de que termine la fase de codificación, sino como una actividad que debe abarcar todo el proceso de desarrollo y mantenimiento, siendo en sí mismo una parte importante en la construcción actual del producto.

Líneas de Investigación, Desarrollo e Innovación

Las principales líneas de investigación son:

- Trabajar con la metodología ágil DDT (Design-Driven Testing). Tomando como escenarios las especificaciones de casos de uso generadas a través de las plantilla CUPIDo, adaptada a esta etapa del desarrollo de software. Realizar pruebas basadas en estos tests.
- Identificación de características inherentes al desarrollo global de software, para la elaboración de los modelos de casos de uso, para que sean tenidas en cuenta en la plantilla de especificación de los casos de uso. Esto se realiza a través de 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.

Resultados y Objetivos

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 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, particularmente testing y también 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.

Se presentaron, en el marco de los proyectos, 2 trabajos en WICC 2014, 2 trabajos de tesis de alumnos en CONAIISI 2014, 1 trabajo en JAIIO-ASSE 2014, 1 trabajo en CACIC 2014, 1 artículo en la revista Te&eT Nº12, 1 artículo en la revista Jóvenes Investigadores de San Juan, 1 Conferencia en Ciclo de Conferencias sobre Ingeniería de Software-FCEFN (proy SPU), participación de un alumno adscripto en TAP Chilecito, ganador y mención honorífica, participación en TAP de la FCEyN-UBA.

Formación de Recursos Humanos

El grupo de investigación que se integran a estas líneas de investigación son:
4 profesores investigadores UNSJ.
5 adscriptos (2 egresados y 3 alumnos)

A la fecha se tiene:
2 tesis de grado finalizadas (2013-2014)
4 tesis de grado a punto de finalizar (2015)
1 beca de Estudiantes Avanzados UNSJ (2014-2015)
1 estancia en UNSJ de 3 meses doctorando de U.Cartagena (2014).
3 integrantes realizaron curso de posgrado "Ingeniería Inversa".
1 curso sobre "Linux".
1 Asistencia a IJCAI School 2014 (SADIO).
2 Asistencia a JAIIO 2014.
1 curso "Creación de Lenguajes específicos de Dominio". UN La Plata. 2014

Referencias

[1] Lund, María Inés, Ferrarini, Cintia, Aballay, Laura Nidia, Romagnano, María Gema, y Meni, Ernesto, «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.

[2] H. B. Christensen, *Flexible, Reliable Software: Using Patterns and Agile Development*. CRC Press, 2011.

[3] S. Chavez, A. Martín, N. R. Rodríguez, M. A. Murazzo, y A. Valenzuela, «Metodología AGIL para el desarrollo SaaS», presentado en XIV Workshop de Investigadores en Ciencias de la Computación, 2012.

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

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

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] «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].

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.° 3, pp. 7-18, jun. 2006.

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

[20] 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.

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

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

[23] B. Homes, *Fundamentals of Software Testing*. John Wiley & Sons, 2013.

[24] T. A. Majchrzak, *Improving Software Testing: Technical and Organizational Developments*. Springer, 2012.

[25] M. Stephens y D. Rosenberg, *Design Driven Testing: Test Smarter, Not Harder*. Apress, 2010.