Validación de Requerimientos a través de Modelos Conceptuales - Modelos y Transformaciones

Marcelo Marciszack, Ramiro Pérez, Claudia Castro

Dpto. Ingeniería en Sist. de Información/ Facultad Regional Córdoba/ Universidad Tecnológica Nacional

{ marciszack, ramipez, ingclaudiacastro }@gmail.com

Resumen

El trabajo presentado en este articulo tiene como objetivo implementar una herramienta que permita gestionar y validar requerimientos de software, que ayudará a definir los límites del sistema al momento de formular los requerimientos, controlar y optimizar los procesos, y proveerá al grupo de desarrollo una base para la estimación del tiempo y costo del desarrollo de sistemas de software, permitiendo así conocer el estado del proyecto y el impacto de los cambios en caso de ser requeridos.

Palabras clave: validación, requerimientos, casos de uso, modelos, conceptual, especificación, trazabilidad, software.

Contexto

El presente proyecto se encuentra consolidado dentro de la línea de investigación de Sistemas de Información en el Dpto. de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional Córdoba.
El mismo busca dar solución a uno de los principales problemas de la ingeniería en sistemas relacionado a la elicitación y especificación de requerimientos, que vincula las distintas etapas del proceso de desarrollo de software manteniendo la trazabilidad de estos requerimientos hasta la validación e implementación de los mismos.

Objetivos Generales

Los objetivos Generales del proyecto de investigación son los siguientes:

- Establecer marco teórico metodológico de técnicas para verificar y validar especificaciones de requerimientos de software / casos de uso.
- Construir e implementar una herramienta de software para la gestión de requerimientos/casos de uso, haciendo especial énfasis en la validación de los mismos.
- Definir propuesta metodológica para la especificación de casos de uso.

Objetivos Específicos

- Validación de modelos conceptuales en la especificación de Requerimientos.
- Posibilitar en todo momento la trazabilidad de requerimientos desde su captura, modelado y seguimiento.
- Validación de Requerimientos Funcionales.

Introducción

La actividad de análisis, diseño y construcción de sistemas de información involucra básicamente a tres tipos de actores: los desarrolladores, que codifican los programas en un lenguaje de programación determinado, los analistas, que especifican la funcionalidad que debe tener el sistema resultante y los usuarios, que poseen requerimientos acerca de lo que debería hacer el sistema para satisfacer sus

necesidades de información para la toma de decisiones.

El análisis de requisitos es la fase más importante en el desarrollo de un proyecto software, ya que de un correcto análisis dependerá la correcta implementación de la aplicación [1].

El documento de especificación de requisitos de software supone una especie de contrato entre usuario y desarrolladores en el que unos indican sus necesidades, mientras que los otros se limitan a implementar lo que se indica en el documento.

La tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación y, por tanto, el desarrollador y el cliente tienen un papel activo en la obtención de estas necesidades [2]. Las últimas tecnologías utilizadas para la obtención de requisitos permiten una mejor comprensión de los documentos de especificaciones, que hasta ahora eran demasiado técnicos para la correcta comprensión por parte del usuario.

Estas técnicas modernas son los casos de uso, que forman parte del UML (Lenguaje Unificado de Modelado) [3]. Ésta es la principal herramienta utilizada para el diseño completo de proyectos software orientado a objetos. Los casos de uso modelan el sistema desde el punto de vista del usuario, permitiéndole así la comprensión completa del futuro sistema.

Finalmente, se debe indicar que esta fase es posiblemente la más costosa (temporalmente) en el desarrollo de un producto software. Esto se debe a que, en general, el cliente no sabe realmente lo que quiere y requiere la ayuda de los analistas para concretar las funciones que realmente se han de implementar. Por tanto, de la calidad del documento de ERS (Especificación de Requerimientos de Software) dependerá el desarrollo y calidad del producto final. La existencia de un estándar, como es el presentado en este proyecto, para la ERS [4] permite la coherencia en la especificación de requisitos y ayuda a no dejar cabos sueltos.

Los principales objetivos que se identifican en la especificación de requisitos software son:

1. Ayudar a los clientes a describir claramente lo que se desea obtener mediante un determinado software: El cliente debe participar activamente en la especificación de requisitos, ya que éste tiene una visión mucho más detallada de los procesos que se llevan a cabo.

2. Ayudar a los desarrolladores a entender qué quiere exactamente el cliente: En muchas ocasiones el cliente no sabe exactamente qué es lo que quiere. La ERS permite al cliente definir todos los requisitos que desea y al mismo tiempo los desarrolladores tienen una base fija en la que trabajar. Si no se realiza una buena especificación de requisitos, los costes de desarrollo pueden incrementarse considerablemente, ya que se deben hacer cambios durante la creación de la aplicación.

3. Servir de base para desarrollos de estándares de ERS particulares para cada organización: Cada entidad puede desarrollar sus propios estándares para definir sus necesidades. Una buena especificación de requisitos software ofrece una serie de ventajas entre las que destacan el contrato entre cliente y desarrolladores (como ya se ha indicado con anterioridad), la reducción del esfuerzo en el desarrollo, una buena base para la estimación de costes y planificación, un punto de referencia para procesos de verificación y validación, y una base para la identificación de posibles mejoras en los procesos analizados.

Las características deseables para una buena especificación de requisitos software que se indican en el IEEE son las siguientes:

- Correcta
- No ambigua
- Completa
- Verificable
- Consistente
- Clasificada
- Modificable
- Explorable

• Utilizable durante las tareas de mantenimiento y uso

Durante el proceso de validación de requerimientos, se deben llevar a cabo verificaciones sobre requerimientos en el documento de requerimientos [5].

Estas verificaciones comprenden:

1. Verificaciones de validez.
2. Verificaciones de consistencia.
3. Verificaciones de integridad.
4. Verificaciones de realismo.
5. Verificabilidad.

Pueden utilizarse varias técnicas de validación de requerimientos, sin embargo nuestro trabajo se centrará en la construcción de modelos conceptuales basados en los requerimientos para poder realizar su validación [6], [7], [8].

Entre algunas de las técnicas de validación de requerimientos existentes se pueden mencionar las siguientes:

1. Revisiones de requerimientos.
2. Construcción de prototipos
3. Generación de casos de prueba.
4. Análisis de consistencia automático

La validación de requerimientos es importante debido a que los errores en el documento de requerimientos pueden conducir a importantes costos al repetir el trabajo cuando son descubiertos durante el desarrollo o después de que el sistema esté en uso. El costo de arreglar un problema en los requerimientos haciendo un cambio en el sistema es mucho mayor que reparar los errores de diseño o los de codificación.

Principios generales a utilizar en la validación:

1. Especificación de los requisitos.

Una especificación documentada de los requisitos del software proporciona la base para la validación y la verificación. El proceso de validación del software no puede completarse sin un establecimiento de las especificaciones de los requisitos.

2. Prevención de defectos.

Las necesidades del aseguramiento de la calidad del software fijan su atención en la prevención de defectos en el proceso de desarrollo del software y no en probar la calidad del código del software después que se escribe. La prueba del software está muy limitada en su capacidad de detectar todos los defectos latentes en el código del software. La complejidad de la mayoría del software impide que sea probado exhaustivamente. La prueba del software es una actividad necesaria.

3. Tiempo y esfuerzo.

La validación del software requiere tiempo y esfuerzo. La preparación para la validación debe comenzar con anticipación; es decir, durante la planificación del diseño y desarrollo y el diseño de la entrada de datos. La conclusión final que muestra que el software se encuentra validado debe estar basada en la evidencia recolectada a partir de los esfuerzos planificados dirigidos a lo largo del ciclo de vida del software.

4. Ciclo de vida del software.

La validación del software tiene lugar dentro del ambiente del ciclo de vida establecido del software. El ciclo de vida del software contiene las tareas de ingeniería de software y la documentación necesaria para soportar la validación del software. Además, el ciclo de vida del software contiene las tareas específicas de verificación y validación que son apropiadas para el uso previsto del software. En la presente nota técnica no recomendamos un modelo particular de ciclo de vida (modelo lineal secuencial, modelo de construcción de prototipos, modelo de desarrollo rápido de aplicaciones, entre otros), sólo establecemos que se deben seleccionar los modelos más apropiados a utilizar en el proyecto de desarrollo del software. Varios modelos del ciclo de vida del software son definidos en la ingeniería del software.

El ciclo de vida puede ser seguido completamente o tener variaciones en su desarrollo, debido a las propias características y naturaleza del software que

se desea desarrollar y su dominio de implementación.

5. Planificación.

El proceso de validación del software se define y controla a través de un plan. El plan de validación del software define lo que será logrado a través del proceso de validación del software.

6. Procedimientos.

El proceso de validación del software se realiza a través del uso de procedimientos documentados. Estos procedimientos establecen "cómo", "quién" y "cuándo" se llevará a cabo la validación del software. Los procedimientos deben identificar las acciones específicas o la sucesión de acciones que deben tomarse para completar las actividades individuales de validación.

7. Validación del software después de un cambio.

Debido a la complejidad del software, un cambio aparentemente pequeño puede tener un impacto significativo en todo el sistema. Cuando se hace cualquier cambio al software (incluyendo pequeños cambios), el estado de validación del software necesita ser restablecido. Siempre que el software sea cambiado, un análisis de validación debe dirigirse no solamente para la validación del cambio individual, sino también para determinar la magnitud e impacto de ese cambio en la totalidad del software.

8. Alcance de la validación.

El alcance de la validación debe estar basado en la complejidad del software y en los riegos de seguridad; no en el tamaño de la organización o la restricción de los recursos. La selección de las actividades y tareas a llevar a cabo durante la validación deben corresponderse con la complejidad del diseño del software y el riesgo asociado con la utilización del software para el uso previsto. La documentación de la validación debe ser suficiente para demostrar que todos los planes y procedimientos de validación del software se han completado con éxito.

9. Independencia de la validación.

Las actividades de validación deben ser conducidas cumpliendo el precepto básico de gestión de la calidad sobre la "independencia de la revisión". La auto-validación es sumamente difícil.

Cuando sea posible, siempre es mejor llevar a cabo el proceso de validación de manera independiente, sobre todo para las aplicaciones que poseen un alto riesgo. Algunas organizaciones optan por la contratación de una tercera parte independiente, pero esta solución no siempre es factible. Otro enfoque es asignar la validación a miembros del personal de la propia organización que no están involucrados en el desarrollo del software o en su aplicación, pero que tienen suficiente conocimiento para evaluar el proyecto y conducir el proceso de validación. En estos casos las organizaciones más pequeñas necesitan ser creativas en la disposición y asignación de las tareas para mantener en todo momento la independencia.

10. Flexibilidad y responsabilidad.

La aplicación específica de estos principios de validación del software puede ser muy diferente de una aplicación a otra. El fabricante o desarrollador del software tiene flexibilidad a la hora de escoger como aplicar estos principios de validación, pero es el responsable de demostrar que el software se ha validado. Las etapas del proceso de validación descritas aquí pueden ser simplificadas dependiendo de la naturaleza del software y de su uso previsto.

Líneas de investigación y desarrollo

Para el modelado de requerimientos se selecciono la Notación para el Modelado de Procesos de Negocio (BPMN por sus siglas en ingles Business Process Modeling Notation) [9] es una notación grafica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (workflow).

Haciendo uso de la capacidad de BPMN, de modelar procesos como flujos, extendemos su uso para describir los sub

procesos que el sistema debe realizar, podemos aprovechar las ventajas que nos brinda esta herramienta.

Se plantearon 2 niveles de modelado, el primero a nivel de procesos de negocio, como lo plantea originalmente BPMN. Y luego un nuevo nivel, a nivel de procesos de sistema de información, usando la misma notación, y con relación a los procesos de negocio que da soporte. Una capa adicional podría añadirse, a nivel de detalle de implementación, pero ya cae fuera de los alcances del proyecto de investigación.

Para alcanzar mapeo y trazabilidad entre la capa de negocio y la de sistema de información, evaluamos distintas herramientas, como QVT, MOF, y XSLT. Finalmente usamos transformaciones XSLT, ya que el formato estándar de BPMN es XPDL, el cual está basado en XML; asimismo no encontramos disponibles herramientas para QVT usando BPMN. Otro inconveniente que se presento fue la curva de aprendizaje para usar QVT o MOF, a diferencia de XSLT que es bastante sencillo de aprender y usar.

De esta manera podemos tener requerimientos del sistema de información modelados de manera de workflows, que dan soporte a los procesos de negocio.

Aplicando transformaciones XSLT a los archivos XPDL podemos disponer de diagramas de casos de uso, de actividad y de clases, en formato XMI.

Al mismo tiempo, se hizo un trabajo anexo usando casos de uso, principalmente usados en UML, dada la amplia aceptación como medio para describir las interacciones realizadas entre el sistema y su entorno. [10]

El Lenguaje Unificado de Modelado (UML) define un caso de uso como "la especificación de una secuencia de acciones, incluyendo variantes, que un sistema (o un subsistema) pueden llevar a cabo, en interacción con los actores del sistema " [11]

Actualmente UML es un estándar altamente adoptado que define los conceptos centrales para el modelado de casos de uso. Al aplicar este estándar, los casos de uso se identifican y estructuran mediante los diagramas de casos de uso de UML y luego se especifican con descripciones textuales. [10]

Usando casos de uso, se encontró que no están expresamente especificados los atributos de un caso del uso. Además, una de las cosas más sorprendentes sobre el estándar UML es la falta de detalle sobre la estructura de los casos de uso. [12]

De esta manera, se observo la necesidad de contar con una herramienta que permita definir y administrar los casos de uso de un sistema software, considerando los atributos comunes y mínimos a distintos autores. Además, se investigo sobre la relación de los casos de uso, y las distintas versiones de un sistema software.

Distintas versiones del sistema, pueden tener más casos de usos agregados que la versión anterior, pueden tener casos de usos modificados o deprecados. Para ello es clave también contar con información de la relación de los casos de uso con cada una de las versiones del sistema, garantizando integridad de producto, trazabilidad y proporciona medios para agilizar la auditoría.

Resultados y Objetivos

Para la línea de BPMN, fue posible realizar un modelo a nivel del sistema de información a partir del modelo de procesos de negocios, mediante transformaciones XSLT. También fue posible realizar transformaciones del modelo XPDL a archivos para simuladores de autómatas finitos.

En el corto plazo seguiremos modificando las transformaciones XSLT con el objetivo de usarlos con SMT Solver, redes de Petri o con autómatas finitos, para poder analizar el modelo, y su validez.

Así, para la primera versión de la herramienta de gestión de requerimientos usando UML, consideramos la administración de sistemas, sus versiones,

casos de uso de una versión del sistema, en conjunto con la administración de usuarios y el historial de modificaciones de cada componente por parte de los usuarios,

Las principales funcionalidades que provee la herramienta web de casos de uso se describen a continuación:

- Administración de los atributos de un sistema y versiones.
- Administración de los atributos de casos de uso, incluyendo precondiciones, pasos, alternativas, etc.
- Diseño del Modelo Conceptual.
- Gestión de cambios en los requerimientos.
- Clasificación los requerimientos.
- Priorización de los requerimientos.
- Trazabilidad de los requerimientos.
- Validación del Modelo Conceptual.
- Visualización de requerimientos.
- Generación de reportes y exportación de los modelos a distintos formatos como XML, PDF, XMI, etc.
- Gestión de la configuración de los requerimientos.

Cabe destacar que haremos especial énfasis en la validación de los requerimientos, pero también abarcaremos los demás aspectos de la gestión de los mismos para lograr una herramienta integral que permita hacer el seguimiento continuo de su evolución.

Formación de Recursos Humanos

Este proyecto prevé la formación de recursos humanos que formen parte del mismo, y que al participar del proyecto contribuyan en su formación.

El mismo cuenta con dos becarios de investigación, que forman parte del presente proyecto, y que serán de ayuda en la recolección, manipulación y tabulación de la información de entrenamiento del sistema.

Actualmente participan 3 tesistas de la maestría de Ingeniería en Sistemas de Información y un doctorando de Ingeniería de Software que desarrollan sus trabajos de en el ámbito del proyecto, lo cual permitirá una contribución a su formación académica.

Al mismo tiempo se prevé vinculaciones con otras redes de investigación; al igual que la realización de un acuerdo de transferencia de resultados con algún organismo que desee considerar los resultados producidos por el sistema que se desarrollará.

Referencias

[1] I. Sommerville. Software Enginnering, Computing Departament, Lancaster University, John Willey & Sons Ltd., 2005.

[2] A. Davis Software requeriments Object, functions and states; Pretince Hall international Inc. 1993.

[3] J. Rumbaugh, I. Jacobson, G.Booch "The Unified Modelling Language Reference" Addisson Wesley 1999.

[4] ANSI/IEEE Standard 830-1984: Standard for software Requeriments Specifications, The institute of Electrical and Electronic Engeniers, New York, 1984.

[5] M. Jackson " Software Requeriments & Specifications". Addisson Wesley.1995.

[6] G. Kotonya, I. Sommervivielle. Requeriments Engineering, Process and Tecniques. John Wiley & sons. 1998.

[7] Emilio Insfrán, Isabel Díaz y Burbano Margarita. Modelado de Requisitos para la Obtención de esquemas conceptuales. Disponible en: http://www.dsic.upv.es/~einsfran/papers/39-ideas2002.pdf

[8] C. Leonardi, J.C.S. Leite, Gustavo Rossi. "Una estrategia de Modelado Conceptual de Objetos , basada en Modelos de requisitos en lenguaje natural ".Tesis de Maestría Universidad Nacional de la Plata.

[9] J. D. Pérez. "Notaciones y lenguajes de procesos. Una vision global". Tesis de Doctorado Universidad de Sevilla.

[10] V.E. Saldaño. "Descubrimiento de Servicios Geográficos a partir de casos de

uso textuales". XV Congreso Argentino de Ciencias de la Computación.
[11] S. Somé. "A Meta-Model for Textual Use Case Description". J. of Object Technology, 8(7).
[12] D. Coleman. "A Use Case Template: draft for discussion". Fusion Newsletter.