Trazabilidad de Versiones en Ingeniería de Requisitos

Andrea F. Vera[(1)], Graciela D. S. Hadad[(1)], Jorge H. Doorn[(1,2)]
[(1)]DIIT, Universidad Nacional de La Matanza, Florencio Varela 1903, San Justo
[(2)]INTIA, Dto de Computación y Sistemas, Facultad de Ciencias Exactas, UNCPBA, Pinto 399, Tandil
{avera, ghadad}@ing.unlam.edu.ar, jdoorn@exa.unicen.edu.ar

Resumen

La trazabilidad persiste como un aspecto relevante y difícil de tratar en la práctica de la Ingeniería de Software. El presente proyecto considera tratar en forma combinada la trazabilidad y el manejo de versiones en el caso de modelos de requisitos. Es un hecho observable que las trazas fluyen esencialmente del proceso del negocio hacia los requisitos, mientras que los cambios de versiones suelen tener origen en fenómenos casi independientes de este flujo. Esta ortogonalidad puede ser muy bien descripta mediante un modelo conjunto de trazas y versionado. Este modelo debe ser acompañado de guías de producción de trazas para modelos basados en lenguaje natural. La idea primordial es que el modelo de trazas esté desacoplado de los modelos del problema y que la producción de trazas se realice simultáneamente con el modelado de requisitos y en forma automática o semi-automática. Por otro lado, se manejan los cambios en los modelos de requisitos generando concurrentemente cambios en las trazas, versionando luego las trazas.

Palabras clave: gestión de requisitos, modelos de requisitos, trazabilidad, sistema de versionado.

Contexto

Esta propuesta es parte del proyecto de investigación “Adaptabilidad y Completitud en Procesos de Requisitos” del Departamento de Ingeniería e Investigaciones Tecnológicas de la UNLaM.

Introducción

La importancia del adecuado conocimiento de los requisitos de un sistema de software y su impacto sobre todo el proceso de desarrollo ha sido estudiada a lo largo de varias décadas. Muchas y muy importantes contribuciones se han producido como consecuencia de una gran cantidad de esfuerzos de investigación, de actividades de normalización y de aplicación práctica de modelos, normas y heurísticas disponibles.

El proceso de la ingeniería de requisitos debería incluirse en los modelos de procesos de software. Por ejemplo, el proceso de requisitos propuesto por Leite et al. [Leite 04] consiste en, primero, la construcción de un glosario denominado Léxico Extendido del Lenguaje (LEL) y de un conjunto de escenarios actuales (EA) que representan situaciones observables en el universo de discurso; segundo, construir escenarios que representan situaciones proyectadas con el nuevo sistema del software, denominados escenarios futuros (EF); y finalmente, definir las especificaciones de requisitos del sistema de software (ER) en base al conocimiento adquirido y modelado en las etapas previas.

Dada la evolución de los requisitos, es indispensable gestionar los mismos [Leite 97]. Ello involucra básicamente la administración de las dependencias entre los mismos y la administración de las vinculaciones entre el documento de requisitos y otros documentos, modelos y componentes del software, como también hacia sus orígenes [Palmer 97] [Kotonya 98] [Davis 99]. La trazabilidad de los requisitos es un tópico de la gestión de requisitos que se encarga de mantener la evolución de estos a través del ciclo de vida del software en ambas direcciones: hacia adelante en las siguientes etapas del proceso de desarrollo y mantenimiento, y hacia atrás hasta sus orígenes. La trazabilidad de requisitos no solo concierne con la gestión de cambios

sino también ayuda en la verificación y validación de requisitos, en el control del proceso de software [Davis 99] [Palmer 97] y en la detección de conflictos, asegura la consistencia entre decisiones tempranas y tardías, y permite controlar la alocación de requisitos, entre otras facilidades.

A pesar de los múltiples propósitos que cubre la trazabilidad de requisitos, a menudo se aplica parcialmente en la práctica. Esto no solo se debe a los altos costos de producción y mantenimiento de las trazas, sino también al esfuerzo de definir las trazas necesarias.

Los problemas en la trazabilidad, que aún persisten se deben principalmente a la falta de herramientas automáticas o semi-automáticas para identificar y mantener trazas [Cleland-Huang 03] [De Lucía 07]. Dado que la recolección y mantenimiento de la información de rastreo es de muy alto costo, se deben tener políticas que indiquen qué tipo de rastreos se realizarán y cómo se mantendrá dicha información.

El trabajo de Huffman Hayes et al. [Huffman Hayes 07] se basa principalmente en la recuperación automática de trazas para generar una matriz de trazabilidad de requisitos. Se utiliza una herramienta que produce vínculos candidatos entre dos artefactos textuales, permitiendo que manual-mente se eliminen vínculos y se agreguen nuevos. Antoniol et al. [Antoniol 02] y Marcus & Maletic [Marcus 03] han presentado propuestas similares, permitiendo la generación automática de trazas desde artefactos textuales hacia código fuente, mientras que la propuesta de Cleland-Huang et al. [Cleland-Huang 05] genera automáticamente vínculos entre artefactos gráficos (Grafo de Interdependencia de Softgoals y diagramas UML). En todas estas propuestas las trazas se producen desde artefactos ya existentes. Estos autores proponen la generación de vínculos en tiempo de ejecución, es decir, el vínculo se crea cuando se necesita recuperar una traza. Esta modalidad evita el constante mantenimiento de vínculos debito a la evolución, aunque afecta la velocidad de recuperación de trazas.

Líneas de investigación y desarrollo

En este proyecto se detallará un mecanismo de generación de trazas que acompañe a cada actividad del proceso de requisitos junto con un mecanismo de versionado de trazas, el cual permitirá mantener la historia de los cambios. Para ello, se refinará el modelo de trazas desarrollado inicialmente [Doorn 10].

Este modelo de trazas apoya el proceso de requisitos de Leite et al. [Leite 04]. Las ER sirven de ancla para la pre y post trazabilidad, permitiendo el rastreo de los requisitos hacia sus orígenes (vinculando las ER con los EF, estos con los EA y con los símbolos del LEL) y hacia delante a artefactos externos al proceso de requisitos (modelos de diseño, código, casos de prueba, entre otros).

Un tema relevante es definir y seleccionar el tipo de traza a utilizar. Existen dos tipos de trazas posibles: trazas incrementales y trazas largas (Figura 1). Una traza larga puede ir directamente de una fuente de información (FI) a un EA (abarca la FI, el símbolo del LEL y el EA), mientras que una traza incremental para llegar desde una FI a un EA, debe vincularse desde la FI al símbolo del LEL, y desde éste vincularse al EA.

El proceso de requisitos utilizado [Leite 04] admite el uso de cualquiera de estos tipos de trazas. Una desventaja que tienen las trazas largas es su mantenimiento,

debido a la gran información que contienen. Por ejemplo, una traza que vincula una FI con un EF va a contener información de los vínculos de la FI, del símbolo del LEL, del EA y del EF en cuestión. Por otro lado, su ventaja está dada por el acceso directo y rápido que presenta de un elemento trazable a otro elemento. Obviamente, estas bondades se invierten al usar trazas incrementales.

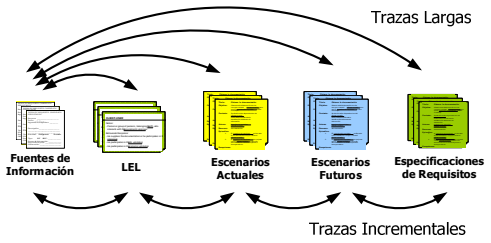

Figura 1. Tipos de Trazas

En el campo del versionado, existe una gran variedad de estrategias [Sommerville 05], que abarcan desde la forma de generación de versiones hasta el mantenimiento y control de cada cambio. Por un lado, se debe decidir el nivel de especificidad de las versiones: por cada elemento individual del modelo, por grupos de elementos, por el modelo íntegro, e incluso por un conjunto consistente y cohesivo de modelos. Es decir, se debe determinar el tipo de ítem a versionar. La estrategia debe identificar la frecuencia de emisión de una versión: desde una nueva versión por cada cambio efectuado, o el acumulado de sucesivos cambios hasta la decisión de una nueva versión del ítem. La estrategia debe determinar cómo guardar cada cambio realizado sobre cada elemento, y para ello existen varias formas de almacenamiento de cambios:

- versión vieja + Δ cambio
- versión vieja y versión nueva
- versión nueva - Δ cambio

El obstáculo de la primera opción es la dificultad en la legibilidad del estado actual pero aporta una buena visibilidad de los cambios. La segunda opción tiene buena legibilidad de ambos estados pero sin reconocer los pasos transitados. La tercera opción presenta el estado actual sin problemas y también buena visibilidad de los cambios, yendo hacia atrás pero sin saber hasta donde retroceder. Debe tenerse en cuenta también que llevar una historia de cambios para reconstruir versiones hacia atrás o hacia delante es una opción costosa por la complejidad que ello involucra. La segunda opción puede mejorarse llevando una historia simple de cambios, sin requerir a partir de ella reconstrucciones de versiones. Es decir, que la traza permite reconocer el contexto y los motivos del cambio sin la complejidad del registro de todos los detalles que involucra el mismo. Debe analizarse la relación beneficio-costo de cada opción para cada proyecto en particular.

En el caso de modelos en lenguaje natural, la segunda opción de bajo costo comparativo es recomendable, porque las trazas se pueden manejar en forma simple. Una traza puede informar la causa de un cambio, quién y cuándo se realizó y los efectos producidos, por ejemplo: lista de símbolos del LEL, de EA, de EF y/o de requisitos afectados (ER).

Resultados y Objetivos

El objetivo propuesto es definir un modelo de trazas que permita seguir la traza de los requisitos a lo largo de todo el proceso de requisitos y que, a su vez, sea extensible a todo el ciclo de vida del software y proveer un modelo de

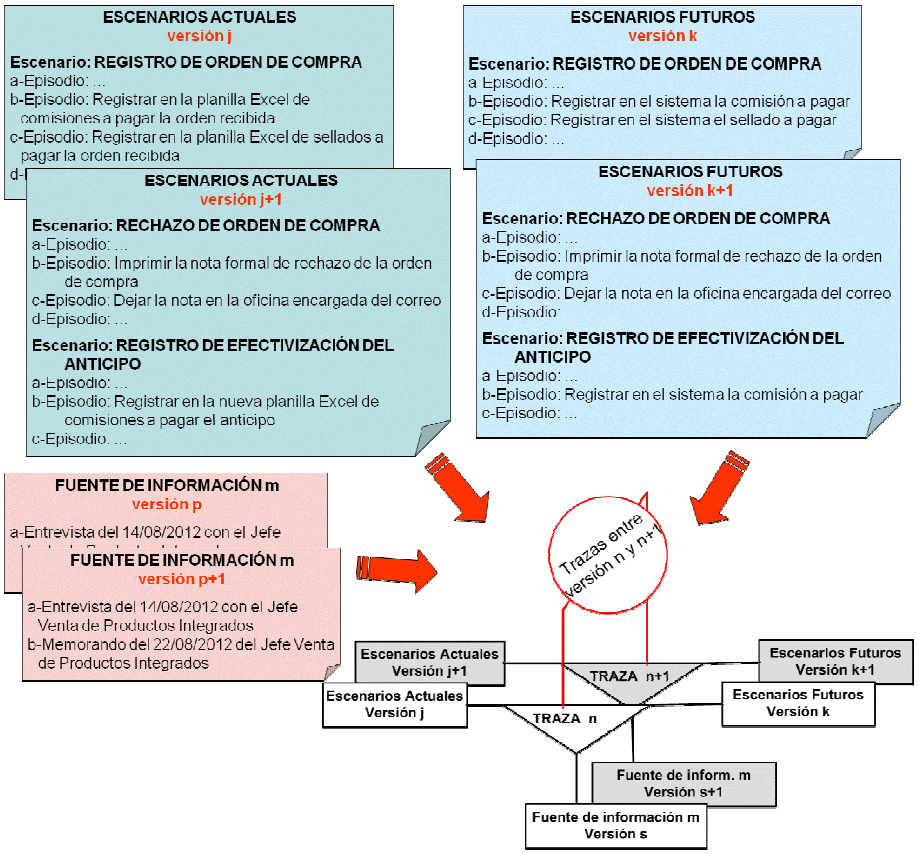

Figura 2. Modelo de Versionado de Trazas

versionado. Como aspecto distintivo de otras estrategias de trazabilidad, la propuesta presentada involucra un modelo de trazas independiente de los modelos de requisitos. Los objetivos específicos son los siguientes:

- Refinar el modelo de trazas.
- Refinar el proceso de captura de trazas.
- Diseñar en detalle el proceso de navegación con trazas.
- Refinar el concepto de trazado de versiones adoptado.

Se ha definido un esquema del modelo de versionado de trazas, que se presenta en la Figura 2. La Traza contiene información acerca de los elementos modelables del proceso de requisitos (ítems) que se relacionan. La Traza entre Versiones relaciona dos Trazas, donde una Traza se ha generado a partir de la otra por un cambio en uno o más de los ítems que relacionaba; por ende, la Traza entre Versiones contiene la causa o motivo que ha generado la creación de la nueva versión de Traza. Se ha definido en forma preliminar el contenido de las entidades y atributos del modelo de trazas y versionado:

Ítem

<id tipo ítem, id ítem, referencia ítem>

Traza

<id traza, $\{\text{id tipo ítem, id ítem}\}_2^N$, motivo>

Traza entre versiones

<id traza N, id traza N+1, motivo>

siendo: id tipo ítem = {símbolo LEL, EA, EF, ER, FI}

Se planifica estudiar en detalle cada uno de los sub-procesos involucrados en el proceso de requisitos desde el punto de vista de las trazas para proveer mecanismos automáticos o semi-automáticos de registro de las mismas.

Algunas de las tares de investigación ya realizadas consistieron en la definición de guías para producir trazas en cada actividad del sub-proceso de extracción de requisitos desde los EF.

Formación de Recursos Humanos

El presente proyecto es parte de la tesis doctoral "Modelado del registro de trazas en la Ingeniería de Requisitos" que está desarrollando la Ing. Andrea F. Vera en la UNLP.

Referencias

[Antoniol 02] Antoniol G., Canfora G., Casazza G., De Lucia A., Merlo E., "Recovering Traceability Links between Code and Documentation", IEEE Transactions on Software Engineering, 28(10):970-983, 2002.

[Cleland-Huang 03] Cleland-Huang J, Chang CK, Christensen M, "Event-based traceability for managing evolutionary change", IEEE Trans. Softw. Eng., 29(9):796–810, 2003.

[Cleland-Huang 05] Cleland-Huang J., Settimi R., BenKhadra O., Berezhanskaya E., Christina S., "Goal-Centric Traceability for Managing Non-Functional Requirements", Internacional Conference on Software Engineering, EEUU, pp. 362-371, 2005.

[Davis 99] Davis A., Leffingwell D., "Making Requirements Management Work For You", Crosstalk, The Journal of Defense Software Engineering, 12(4), 1999.

[De Lucía 07] De Lucia A., Fasano F., Oliveto R., Tortora G., "Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods", ACM Transactions on Software Engineering and Methodology, 16(4):13.1-13.50, 2007.

[Doorn 10] Doorn JH, Hadad GDS, Vera AF, "Consolidación de Requisitos: Gestión de Requisitos", Anuario de Investigaciones: Resúmenes Extendidos 2010, Osvaldo Spósitto y Andrés Dmitruk (eds), Universidad Nacional de La Matanza, San Justo, ISBN: 978-987-1635-55-9, pp.19-26, 2012.

[Huffman Hayes 07] Huffman Hayes J., Dekhtyar A., Karthikeyan Sundaram S., Ashlee Holbrook E., Sravanthi Vadlamudi, April A., "REquirements TRacing On target (RETRO): improving software maintenance through traceability recovery", Innovations in Systems and Software Engineering, 3(3):193-202, 2007.

[Kotonya 98] Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques, John Wiley & Sons, 1998.

[Leite 97] Leite, J.C.S.P., "Software Evolution, The Requirements Engineering View", keynote address en anales de XXVI JAIIO - SoST'97 Simposio en Tecnología de Software, Buenos Aires, Argentina, pp.21-23, 1997.

[Leite 04] Leite, J.C.S.P., Doorn, J.H., Kaplan, G.N., Hadad, G.D.S., Ridao, M.N., "Defining System Context usingScenarios", en el libro "Perspectives on Software Requirements", Kluwer Academic Publishers, EEUU, ISBN: 1-4020-7625-8, cap. 8, pp.169-199, 2004.

[Marcus 03] A. Marcus, J. Maletic, "Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing", Twenty-Fifth International Conference on Software Engineering (ICSE), pp. 125-135, 2003.

[Palmer 97] Palmer J.D., "Traceability", en: Software Requirements Engineering, R.H. Thayer, M. Dorfman (eds.), IEEE Computer Society Press, 2º ed., pp. 364-374, 1997.

[Sommerville 05] Sommerville I, "Ingeniería del Software", Pearson Educación, 7º ed., 2005.