Evaluación de Variantes de Inspección en la Ingeniería de Requisitos

Alberto Sebastián[1], Graciela D. S. Hadad [1,2], Tomás Damonte[1], Ariel Vera[2], Ezequiel Robledo[2], Daniela Raffo[1]

[1]Facultad de Ingeniería y Tecnología Informática, Universidad de Belgrano
[2]Escuela de Informática, Universidad Nacional del Oeste
{alberto.sebastian, graciela.hadad, tomas.damonte}@comunidad.ub.edu.ar, avera@uno.edu.ar, ezeroble@hotmail.com.ar, daniela.raffo@comunidad.ub.edu.ar

RESUMEN

Es frecuente durante el proceso de requisitos construir modelos escritos en lenguaje natural, pues facilitan la elicitación, validación y negociación con el cliente. Estas facilidades se incrementan cuando los modelos se escriben usando el lenguaje propio del cliente. Para ello, se suele construir tempranamente un glosario con los términos utilizados en el contexto, denominado Léxico Extendido del Lenguaje. Este permite mejorar la comunicación entre los involucrados, sirviendo de apoyo a la descripción de modelos subsecuentes. Aunque, como todo modelo generado siguiendo heurísticas, presenta habitualmente inconsistencias, errores, omisiones y ambigüedades. Estas últimas aparecen básicamente por el uso del lenguaje natural. Estudios de estimación de completitud sobre este modelo establecieron la ocurrencia de un número significativo de omisiones. Es relevante poder detectar estos defectos y corregirlos oportunamente, evitando su propagación sobre otros modelos. Se han diseñado variantes de la técnica de inspección sobre el modelo léxico, las cuales identifican distintos tipos de defectos aunque su eficiencia y eficacia no son fácilmente comparables dado que no apuntan a detectar los mismos defectos. Se propone un análisis de estas variantes que permita establecer cuál es la más apropiada según el tipo de defecto que se requiera atender y el tiempo disponible.

Palabras clave: Ingeniería de Requisitos, Inspección de Modelos, Completitud de Modelos, Ambigüedad.

CONTEXTO

La propuesta que se presenta es parte de los proyectos de investigación “Gestión de la calidad de un modelo léxico en el proceso de requisitos” de la Universidad de Belgrano y “Tratamiento de los factores situacionales y la completitud en la ingeniería de requisitos” de la Universidad Nacional del Oeste.

1. INTRODUCCIÓN

El proceso de requisitos involucra comprender el comportamiento actual en un contexto y definir la situación futura al incorporar un sistema de software [1]. En este proceso suelen utilizarse modelos escritos en lenguaje natural dado su cercanía a los clientes [2], aún cuando este tipo de modelos presentan defectos, principalmente del tipo ambigüedades y omisiones [3, 4, 5, 6].

Se han realizado varios estudios sobre la completitud de modelos en lenguaje natural en la Ingeniería de Requisitos, arrojando niveles excesivamente altos de omisiones [7, 8, 9]. Otros estudios se centraron en la ambigüedad de estos modelos, tal es el trabajo de Ben Achour et al. [10], quienes a través de un estudio empírico observaron que el 50% de los casos de uso contenían errores de terminología.

Dado que estos modelos construidos en el proceso de requisitos son la base para etapas posteriores del desarrollo de software, es necesario que presenten un nivel de calidad adecuado, pues los defectos no identificados tempranamente se trasladarán en cascada a modelos posteriores. Un mecanismo para

reducir estos defectos es aplicando técnicas de verificación, tales como las revisiones y las inspecciones [11].

La inspección de software [12, 13] fue diseñada como un proceso formal de detección de defectos en código fuente, posteriormente adaptada a modelos. Este proceso se compone de seis pasos bien definidos y con roles específicos en cada paso: inspector, productor (autor), moderador y escriba. Los pasos del proceso son:

1. Planeamiento: identificar el material a inspeccionar y las personas a cumplir con cada rol, y establecer la fecha de reunión.
2. Apreciación global: trasmitir a los inspectores una descripción general del material a inspeccionar.
3. Preparación: leer el material por parte de los inspectores.
4. Reunión: determinar cuáles son defectos en el material por parte de los inspectores y transmitirlos a los productores del mismo, asistidos por el moderador mientras el escriba registra los hechos.
5. Corrección: eliminar los defectos del material revisado, a cargo de los productores.
6. Seguimiento: determinar el estado de la corrección, a cargo del moderador.

En general, en este proceso formal de revisión los defectos son catalogados por tipo y por severidad, y se cuantifican los tiempos de detección, para mejorar la eficiencia y eficacia del proceso mismo.

Las inspecciones han surgido como una de las técnicas de aseguramiento de calidad más eficaces en la ingeniería de software, debido a lo cual se han diseñado diferentes variantes aplicadas a modelos elaborados en el proceso de requisitos [14, 15, 16]. Estas variantes se han clasificado según el modo de detección de defectos en la etapa de preparación [17] en: lectura ad-hoc, lectura usando checklist, lectura usando procedimientos y lectura constructiva.

La lectura ad-hoc se hace en una modalidad desestructurada, donde el inspector recibe muy poco apoyo para encontrar defectos. Esto no significa que los participantes de la inspección no escruten el artefacto sistemáticamente. Sólo significa que ningún plan previo está disponible y todo se deja librado a la experiencia del inspector.

La lectura con checklist le da un apoyo más fuerte al inspector en la forma de una lista de preguntas y controles que tienen que ser contestados y revisados en un cierto orden preestablecido. Este enfoque tiene algunas debilidades, como la falta de ayuda en comprender el artefacto bajo estudio y pobre adaptación a un ambiente de desarrollo dado.

La lectura basada en procedimientos da una guía detallada al inspector sobre cómo encontrar defectos específicos, independizándose de la experiencia del inspector.

La lectura constructiva va más allá que meramente revisar un artefacto y producir una lista de defectos, pues durante la lectura el inspector produce una nueva representación del material bajo estudio, la que se analiza posteriormente para detectar defectos.

En pocos trabajos se reportan evaluaciones sobre qué variante de inspección puede ser más provechosa, y si ellas dependen del modelo a verificar y del tipo de defectos que presentan. En algunos experimentos, se siguió la experiencia del inspector [15, 18] y los resultados mostraron que la importancia de la definición del procedimiento disminuía a medida que aumentaba la experiencia del inspector, y que inspectores novicios podían adquirir rápidamente conocimiento sobre defectos típicos y aspectos de calidad. Paech et al. [18] comprobaron que se detectaban en promedio más defectos usando la técnica de lectura basada en procedimientos frente a la lectura con checklist, aunque acarreaba más tiempo de inspección.

2. LÍNEAS DE INVESTIGACIÓN E DESARROLLO

En un proceso particular de requisitos orientado al cliente [19, 20], donde se crean y utilizan modelos en lenguaje natural para lograr un mayor involucramiento de los clientes, el primer modelo que se construye es el Léxico Extendido del Lenguaje (LEL) [21], con el fin de obtener una comprensión acabada del vocabulario que se utiliza en el

contexto del problema. Este modelo se conforma de un conjunto de términos relevantes en dicho contexto, con sus definiciones, dadas por dos componentes: noción (denotación del término) e impacto (su connotación). El modelo considera la definición de términos sinónimos y homónimos, como también jerarquías de términos (términos genéricos y especializados) [20].

Este modelo debe tener la mayor calidad posible, dado que los restantes modelos harán uso de la terminología definida en él y algunos de ellos pueden derivarse del propio LEL [20]. Para lograr esta calidad, se han propuesto heurísticas y procedimientos para reducir defectos en este modelo.

Las heurísticas se enfocan principalmente en la creación del modelo LEL aportando guías de estilo y de contenido [6, 21, 22]. A pesar de la aplicación de estas heurísticas, el nivel de completitud estimado de este modelo ha sido muy bajo. En un estudio realizado por Doorn y Ridao [7] donde estimaron el tamaño de un modelo LEL a partir de nueve muestras del mismo, obtuvieron en el mejor de los casos un nivel de completitud del 51%. En un estudio posterior realizado por Litvak et al. [23], se confirmaron estos valores, aún cuando las muestras del modelo léxico habían sido construidas con heurísticas más precisas.

Por otro lado, se han diseñado mecanismos específicos para la verificación de este modelo. Los defectos a ser identificados han sido categorizados en: discrepancias (inconsistencias), errores, omisiones y ambigüedades, y calificados según su grado de severidad en: alto, medio y bajo [24].

En proyectos previos, se diseñó una técnica de inspección del modelo LEL basada en procedimientos [25] para guiar la detección de diversos tipos de defectos. En esta variante se completa un conjunto de formularios siguiendo un procedimiento para cada formulario, el cual detalla cómo completar el formulario y cómo identificar defectos a través de la información registrada. Es una variante que facilita la detección de defectos a inspectores novatos, aunque insume un tiempo considerable. Por otro lado, tal cual fue propuesta en [25], no establece el grado de severidad de los defectos ni los tipifica; esto fue realizado posteriormente para facilitar evaluaciones de la misma. Las omisiones que permite detectar son relativamente simples y algunas requieren una gran cantidad de comparaciones semánticas, siendo las ambigüedades tratadas como una subclase de omisiones.

Posteriormente, se elaboró una lista de control (checklist) para utilizar como otra variante de la inspección del LEL. Esta lista de control fue refinada incrementalmente a medida que se probaba en diversos casos. En su última versión, contiene 44 ítems de control, y establece por cada ítem el tipo de defecto a identificar y su severidad.

Más recientemente, se diseñó otra variante de la inspección del LEL, basada en la lectura constructiva [24, 26]. Ésta genera un artefacto intermedio, que son mapas conceptuales, uno por cada término del LEL. Cada oración en la noción y en el impacto del término se representa como una proposición en el mapa conceptual del mismo. Los defectos se identifican a partir del análisis sistemático de cada mapa conceptual y de las relaciones entre mapas, mediante guías de análisis. Esta variante se focaliza en la detección de varios tipos de omisiones y ambigüedades con diverso grado de severidad, permitiendo incluso sugerir términos candidatos omitidos, aunque no se aboca a la detección de otros tipos de defectos. Aún cuando debe construirse un artefacto a partir del modelo LEL, se han obtenido resultados relativamente aceptables, en relación al consumo de tiempo y a la cantidad de defectos detectados [27]. Otra particularidad de esta variante es que sugiere correcciones al modelo LEL.

Cabe notar que estas tres variantes de inspección fueron desarrolladas en forma independiente una de otra, por lo que no todo defecto es detectado por todas las variantes, y algunos defectos son identificados por una sola de ellas, lo que dificulta una comparación efectiva de las variantes.

Se espera poder dar pautas a la gerencia de un proyecto de software sobre qué variante de

inspección puede ser más provechosa según el nivel esperado de calidad, los recursos humanos disponibles y los plazos a cumplir.

3. RESULTADOS OBTENIDOS/ESPERADOS

Se ha llevado a cabo un primer análisis de cada variante respecto a qué defectos comunes identifican (ver Tabla 1), cuáles son identificados por pares de variantes y cuáles son exclusivos de una variante (ver Tabla 2).

Tabla 1. Defectos comunes a las variantes

Defecto	Tipo	Severidad
Términos sin referencia a otros términos	Omisión	Media
Términos no referenciados por ningún término	Omisión	Media
Referencia a término no identificada en la definición de un término	Omisión	Media
Término con más de un verbo principal en cada oración de noción e impacto	Omisión	Baja

Tabla 2. Defectos detectados solo por variante con mapas conceptuales

Defecto	Tipo	Severidad
Vocabulario mínimo redundante (con sinónimos internos)	Ambi-güedad	Media
Uso de término del vocabulario mínimo en lugar de término del LEL	Ambi-güedad	Media
Término del vocabulario mínimo no detectado como sinónimo del LEL	Omisión	Media
Frases omitidas en la noción o impacto del término	Omisión	Baja
Términos con oraciones subordinadas complejas	Ambi-güedad	Alta
Omisión de términos de tipo Verbo (uso frecuente)	Omisión	Alta

Debe tenerse en consideración que la inspección con checklist contiene 44 ítems de control, la inspección basada en procedimientos requiere 10 formularios de detección, y la inspección basada en mapas conceptuales utiliza 13 pasos de detección (además de los pasos de construcción de los mapas). Las variantes con checklist y con formularios detectan principalmente errores y omisiones, mientras que la basada en mapas conceptuales se aboca a ambigüedades y omisiones.

Se ha iniciado un experimento controlado para realizar comparaciones de las tres variantes utilizando modelos LEL generados independientemente para distintos casos de estudio, y considerando los tipos de defectos que detectan, su nivel de severidad, los tiempos que insumen y la experiencia de los inspectores.

Asimismo, se está desarrollando una herramienta de software, denominada Gestor de Inspecciones, implementada en una plataforma web, utilizando lenguaje PHP y base de datos relacional MySql. La herramienta permite administrar los resultados de inspecciones, realizadas por ingenieros de requisitos autorizados, sobre modelos LEL, soportando el almacenamiento de resultados para las tres variantes de inspección. Esta herramienta facilitará el análisis de la eficiencia y efectividad de las distintas variantes de inspección.

Por otro lado, se espera establecer las causas de los defectos detectados, de manera tal de poder definir heurísticas que permitan evitar la ocurrencia de estos defectos, ya sea al elicitar conocimiento del contexto de aplicación o al modelar, es decir, heurísticas que se aboquen a completar información y a escribir el modelo LEL sin ambigüedades.

4. FORMACIÓN DE RECURSOS HUMANOS

En el proyecto de la Universidad de Belgrano participan tres investigadores, uno de ellos en formación y un alumno de la tecnicatura en Programación de Computadoras, cumpliendo su Trabajo Social Profesional, mientras que en el tema de completitud dentro del proyecto de la Universidad Nacional del Oeste participa un investigador con dos alumnos becarios.

5. BIBLIOGRAFÍA

[1] Nuseibeh, B., Easterbrook, S. (2000) Requirements Engineering: A Roadmap. Future of SE Track 2000, pp.35-46

[2] Ryan, K. (1993) The Role of Natural Language in Requirements Engineering, IEEE International

Symposium on Requirements Engineering, San Diego, CA, pp. 240-242.
[3] Zowghi, D., Gervasi, V. (2002). The Three Cs of Requirements: Consistency, Completeness, and Correctness, 8th International Workshop on Requirements Engineering: Foundation for Software Quality, Essen, Germany: Essener Informatik Beitiage, pp. 155-164
[4] Berry, D.M., Kamsties, E. (2004) Ambiguity in Requirements Specification. En: Leite & Doorn (eds.) Perspectives on Software Requirements, pp.7-44. Kluwer Academic Publishers.
[5] Hadad, G.D.S., Litvak, C.S., Doorn, J.H., Ridao, M.N. (2015) Dealing with Completeness in Requirements Engineering. En: Khosrow-Pour, M. (ed), Encyclopedia of Information Science and Technology, Third Edition, pp.2854-2863. IGI Global, Information Science Reference.
[6] Litvak, C.S., Hadad, G.D.S., Doorn J.H. (2017) Nominalizations in Requirements Engineering Natural Language Models. En: Encyclopedia of Information Science and Technology, Fourth Edition, IGI Global.
[7] Doorn, J.H., Ridao, M. (2003) Completitud de Glosarios: Un Estudio Experimental. 6th Workshop on Requirements Engineering.
[8] Ridao, M., Doorn, J. (2006) Estimación de Completitud en Modelos de Requisitos Basados en Lenguaje Natural. 9th Workshop on Requirements Engineering, ISSN: 1413-9014, pp. 151-158.
[9] Hadad, G.D.S., Litvak, C.S., Doorn, J.H. (2014) Problemas y Soluciones en la Completitud de Modelos en Lenguaje Natural. II Congreso Argentino de Ingeniería. ISBN:978-987-1662-51-7.
[10] Ben Achour, C., Rolland, C., Maiden, N.A.M., Souveyet, C. (1999) Guiding Use Case Authoring: Results of an Empirical Study, International Symposium On Requirements Engineering (RE´99), Limerick, Irlanda, IEEE Computer Society Press, 1999, pp.36-43.
[11] Porter, A.A., Votta, Jr., L.G., Basili, V.R. (1995) Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, IEEE Transactions on Software Engineering, Vol. 21, Nº 6, pp. 563-575.
[12] Fagan, M.E. (1976) Design and Code Inspections to reduce Errors in Program Development, IBM Systems Journal, Vol.15, Nº 3, pp. 182-211.
[13] Fagan, M.E. (1986) Advances in Software Inspections, IEEE Transactions on Software Engineering, Vol.12, Nº 7, pp. 744-751.
[14] Fusaro, P., Lanubile, F., Visaggio, G. (1997) A Replicated Experiment to Assess Requirements Inspection Techniques, Empirical Software Engineering, Vol. 2, Nº 1, pp.30-57.
[15] Porter, A.A., Votta, Jr. (1998) Comparing Detection Methods for Software Requirements Inspections: A Replication Using Professional Subjects. Empirical Software Engineering, Vol. 3, Nº 4, pp. 355-380.
[16] Leite, J.C.S.P., Doorn, J.H., Hadad, G.D.S., Kaplan, G.N. (2005) Scenario Inspections. Requirements Engineering Journal, Springer-Verlag, Vol. 10, Nº 1, pp. 1-21.
[17] Regnell, B., Runesom, P., Thelin, T. (1999) Are the perspectives really different? Further experimentation on scenario-based reading of requirements. En: Requirements engineering with use cases - a basis for software development, Reporte Técnico 132, Lund University, pp.141-180.
[18] Paech, B., Denger, C., Kerkow, D., von Knethen, A. (2005) Requirements Engineering for Technical Products: Integrating Specification, Validation and Change Management, capítulo X, Information Science Publishing, Maté & Silva (eds.), Londres, 2005, pp.153-169.
[19] Leite, J.C.S.P., Doorn, J.H., Kaplan, G.N., Hadad, G.D.S., Ridao, M.N. (2004) Defining System Context using Scenarios. En: Perspectives on Software Requirements, Kluwer Academic Publishers. ISBN:1-4020-7625-8,cap.8,pp.169-199.
[20] Hadad, G.D.S. (2018) Uso de Escenarios en la Derivación de Software. Tesis Doctoral, Universidad Nacional de La Plata.
[21] Hadad, G.D.S., Doorn, J.H., Kaplan, G.N. (2009) Creating Software System Context Glossaries. En: Encyclopedia of Information Science and Technology, 2nd Edition, pp.789-794. IGI Global.
[22] Litvak, C.S., Hadad, G.D.S., Doorn, J.H. (2014) Heurísticas para el modelado de requisitos escritos en lenguaje natural, CACIC 2014 - XX Congreso Argentino de Ciencias de la Computación, Buenos Aires, pp. 682-691.
[23] Litvak, C.S., Hadad, G.D.S., Doorn, J.H. (2013) Mejoras semánticas para estimar la Completitud de Modelos en Lenguaje Natural, 1er. Congreso Nacional de Ingeniería Informática / Sistemas de Información, Córdoba.
[24] Sebastián, A., Hadad, G.D.S., Robledo, E. (2017) Inspección centrada en Omisiones y Ambigüedades de un Modelo Léxico, CIBSE 2017 – XX Congreso Iberoamericano en Software Engineering, track Workshop on Requirements Engineering, Buenos Aires.
[25] Kaplan, G., Hadad, G., Doorn, J. Leite, J. (2000) Inspección del Léxico Extendido del Lenguaje. III Workshop on Requirements Engineering, pp.70-91.
[26] Sebastián, A., Hadad, G.D.S. (2016) Enhancing a Lexicon Model by Concept Mapping. En: Computer Science & Technology Series. XXI Argentine Congress of Computer Science – Selected Papers, G.E. Feierherd, P.M. Pesado & C.C. Russo (eds.), Editorial de la Universidad de La Plata (EDULP), pp. 139-151.
[27] Sebastián, A., Hadad, G.D.S. (2016) Experimento Controlado en la Inspección de un Léxico mediante Mapas Conceptuales. III Congreso Argentino de Ingeniería, Resistencia, pp. 2738-2752.