CienciaUAT

ISSN: 2007-7521
cienciauat@uat.edu.mx
Universidad Autónoma de Tamaulipas
México

Ventura Roque-Hernández, Ramón; Díaz-Redondo, Rebeca; Fernández-Vilas, Ana
La especificación de requerimientos de software desde la perspectiva de un nuevo paradigma: los aspectos
CienciaUAT, vol. 6, núm. 3, enero-junio, 2012, pp. 56-59
Universidad Autónoma de Tamaulipas
Ciudad Victoria, México

ISSN 2007-7521. 6(3): 56-59 (Ene-Jun 2012)

La especificación de requerimientos de *software* desde la perspectiva de un nuevo paradigma: los aspectos

Software Requirements Specification from a new paradigm perspective: aspects

Foto: ingimage.com

Ramón Ventura Roque-Hernández[1*], Rebeca Díaz-Redondo[2], Ana Fernández-Vilas[2].
[1]Universidad Autónoma de Tamaulipas. [2]Universidad de Vigo, España.

***Autor para correspondencia: Universidad Autónoma de Tamaulipas. Facultad de Comercio, Administración y Ciencias Sociales de Nuevo Laredo. Ayuntamiento s/n. Col. Infonavit, Nuevo Laredo, Tamaulipas, México. C. P. 88275.**
ramonroque@yahoo.com

RESUMEN

La especificación de requerimientos es una actividad primordial en el desarrollo de sistemas de *software* que orienta en todo momento las acciones del equipo de trabajo; la manera de realizarla varía de acuerdo a la naturaleza del proyecto y a las prácticas adoptadas por los desarrolladores. Una forma relativamente reciente de especificar requerimientos es mediante el paradigma orientado a aspectos: un enfoque que propone una manera diferente de conceptualizar las características importantes del *software*, para administrar mejor la naturaleza transversal de algunas de ellas. Este artículo presenta la problemática de la especificación correcta de requerimientos, contextualiza el paradigma orientado a aspectos en la representación de estas especificaciones y ejemplifica su filosofía con una propuesta que es producto del reciente trabajo de investigación de los autores de este artículo.

PALABRAS CLAVE: especificación, requerimientos, *software*, orientación, aspectos.

ABSTRACT

Requirements Specification is a very important activity in Software Systems Development; it guides the action of the work team at every moment. There are different ways to carry out the requirements specification; the nature of the project and the practices adopted by the developers influence the decision to choose a specific approach. A relatively recent approach is the Aspect-Oriented Paradigm that proposes a different way of conceptualizing the important features of software to better manage the cross-cutting nature of some of these features. This paper discusses the problem of the correct specification of requirements, contextualizes the aspect-oriented paradigm in the representation of requirements specifications, and exemplifies this philosophy with a pro-

Fecha de recepción: 19 de febrero de 2012.
Fecha de aceptación: 4 de junio de 2012.

posal that is the result of recent research work of the authors of this paper.

KEY WORDS: specification, requirements, software, aspect, oriented.

INTRODUCCIÓN

El desarrollo de programas de cómputo involucra varias actividades relacionadas entre sí que conllevan a la obtención de un producto, el cual debe satisfacer una serie de necesidades que originó su creación. Las actividades más relevantes en el desarrollo de programas son el análisis del sistema a realizar, el diseño de una solución, la implementación de esa solución, la realización de pruebas y el mantenimiento del *software* construido. El análisis es la etapa inicial en la cual las necesidades de los usuarios quedan al descubierto y son especificadas por escrito; a esta actividad se le denomina "especificación de requerimientos o requisitos" y tiene una importancia superlativa en el proceso completo de desarrollo porque el documento resultante es un mapa orientador en cada momento del proyecto. Cada requerimiento representa una característica que el *software* debe ser capaz de realizar, por lo que éste debe ser expresado lo más precisamente posible para evitar ambigüedades que conduzcan a re-trabajo inútil y costoso en etapas posteriores. La especificación completa debe, además, ser suficientemente flexible para adaptarse a las necesidades cambiantes del usuario durante el ciclo de vida del proyecto del *software*. La especificación de los requerimientos se puede realizar con diferentes enfoques que involucran desde lenguaje natural hasta lenguajes matemáticos formales, pasando por una gran variedad de puntos intermedios. Las prácticas adoptadas por el equipo de desarrollo y la naturaleza del proyecto de *software* influencian en gran medida la elección de la manera de expresar los requerimientos.

En la presente publicación se destacan la importancia y las dificultades de especificar los requerimientos de cualquier sistema de *software*, se presenta la orientación a aspectos como un paradigma emergente que propone nuevos medios para enfrentar la complejidad en la ingeniería del *software*, se analiza el impacto de este paradigma en la especificación de requerimientos y se expone como ejemplo un modelo aspectual de especificación de requerimientos que promueve la reutilización. Finalmente se presentan las conclusiones de este trabajo.

LA IMPORTANCIA DE ESPECIFICAR LOS REQUERIMIENTOS

Idealmente, los requerimientos de los sistemas deben ser especificados sin ambigüedad antes de seguir adelante en el proceso de desarrollo. Sin embargo, en la realidad existen aplicaciones para las cuales esta especificación nunca se lleva a cabo, o se realiza de manera superficial; estas prácticas impiden fijar metas objetivas, realizar evaluaciones de manera efectiva y llevar un control adecuado de la evolución de los requerimientos durante el ciclo de vida del proyecto (Braude, 2008). Este enfoque informal de desarrollo, generalmente aplicado a proyectos de tamaño pequeño, sería peligroso de trasladar a proyectos grandes, especialmente si del *software* en cuestión depende la seguridad de vidas humanas (Jacky, 2008), como es el caso de los controladores aéreos, ferroviarios o aplicaciones médicas avanzadas, donde se requiere un alto nivel de confiabilidad que se consigue mediante la evaluación y prueba exhaustiva de los requerimientos expresados en algún lenguaje matemático formal.

Una especificación debe contener requerimientos plenamente identificados, que describan de manera clara, precisa y atómica cada característica del sistema. Independientemente de la metodología de desarrollo de *software* utilizada, si estas propiedades no se logran y, además, los requerimientos especificados no resultan ser los correctos, las futuras etapas del desarrollo tendrán inconvenientes y el proyecto completo podría ir directo al fracaso. Los errores que se introducen en la captura de los requerimientos resultan ser los más costosos cuando se detectan en etapas posteriores del proyecto, por eso esta es una actividad delicada de creciente

Fuente de elaboración: adaptada de artículo "Programación orientada a aspectos". Autor: Aldo Marcelo Algorry. Revista USERS. Code No. 3, 2005, págs 60-64.

FIGURA 1

Los intereses transversales afectan varias partes del *software*.
Figure 1. Crosscutting concerns affect software systems in several places.

complejidad (Mathiassen y Tuunanen, 2011) que debe ponderarse en su justa medida.

Una especificación que captura adecuadamente los requerimientos correctos contribuye a que todas las actividades del desarrollo estén bien orientadas hacia la ingeniería de un producto de *software* pertinente, con lo que se establece de antemano un escenario fértil para el éxito del proyecto.

LA ORIENTACIÓN A ASPECTOS COMO PARADIGMA EMERGENTE EN EL DESARROLLO DE *SOFTWARE*

Los nuevos paradigmas de desarrollo de *software* han surgido como respuesta a la necesidad de enfrentar la complejidad de las aplicaciones de una mejor manera con modelos cada vez más evolucionados. Un problema complejo bien identificado en la ingeniería del *software* es la denominada "separación de intereses". Se entiende como interés una parte o unidad que es relevante para un concepto o propósito del procedimiento que se está desarrollando, por ejemplo, la seguridad, la gestión de clientes o el procesamiento de pagos. La separación de intereses se refiere a la manera en la que esas características son conceptualizadas, capturadas y administradas a lo largo del proceso de desarrollo.

La ingeniería de *software* y la industria apun-

tan hacia un paradigma maduro y suficiente para enfrentar la complejidad de la mayoría de los desarrollos de aplicaciones de negocios actuales: el paradigma orientado a objetos. Este modelo, si bien ofrece una manera relativamente sencilla de abstraer conceptos relacionados con las aplicaciones y organizarlos en forma de jerarquías, se queda corto en el manejo de los llamados intereses transversales: aquellos intereses que impactan varias partes del *software*. La fortaleza del paradigma orientado a objetos, radica en la verticalidad de sus abstracciones que promueve la reutilización; su mayor limitación es debida al pobre manejo de la naturaleza horizontal de los intereses transversales (ver figura 1). Es así como con el paradigma orientado a objetos las representaciones de elementos que comparten características comunes (como, por ejemplo, generalizaciones –pago– y especializaciones –pagos con dinero en efectivo, con cheque, con tarjeta de crédito–) se ven ampliamente privilegiadas, mientras que el manejo de intereses transversales (como persistencia, seguridad, manejo de errores, transacciones) da origen a problemas tales como: 1) "intereses dispersos" en donde un mismo interés no puede ser encapsulado en una unidad aislada por lo que queda diseminado por distintas partes del sistema; y 2) "intereses enredados" los cuales ocurren cuando en un mismo interés se atiende una funcionalidad principal y además se incorporan otras competencias adicionales. Estos dos problemas afectan la calidad, la facilidad de entendimiento, mantenimiento y la evolución de las aplicaciones de *software*.

Como una alternativa de solución que busca mejorar las situaciones problemáticas del paradigma orientado a objetos, ha surgido el modelo orientado a aspectos que propone una mejor separación de los intereses al incorporar un nuevo nivel de abstracción para encapsular los intereses transversales y nuevos mecanismos para especificar sus impactos en diferentes partes del programa. La orientación a aspectos surge como tal en 1995, en la fase de implementación con un lenguaje de programación conocido como Aspect-J (Laddad, 2003) que incorporó el término "aspecto". Un aspecto puede definirse como un interés que atraviesa la aplicación, por lo que no puede encapsularse claramente y constituye una unidad conceptual susceptible de ser identificada y separada (Algorry, 2005). Este paradigma ha influenciado otras fases previas del desarrollo como el análisis y diseño, originando el concepto de "aspectos tempranos"; que se refiere a los intereses transversales en las etapas del ciclo de vida previas a la programación como el análisis y diseño (Baniassad y col., 2006).

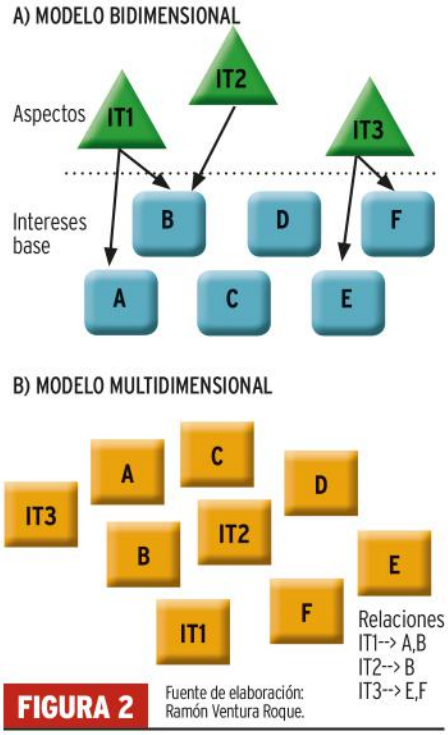

FIGURA 2 Fuente de elaboración: Ramón Ventura Roque.

Comparación entre modelos bidimensional y multidimensional.
Figure 2. Bi-dimensional and Multi-dimensional models.

LA EVOLUCIÓN DE LA ESPECIFICACIÓN DE REQUERIMIENTOS CON EL PARADIGMA ORIENTADO A ASPECTOS

En los últimos años, la manera de realizar especificaciones de requerimientos se ha visto influenciada por los aspectos tempranos. Esta influencia ha consistido principalmente en la incorporación de nuevos modelos de separación de intereses en las especificaciones, desde dos perspectivas de trabajo: la bidimensional y la multidimensional.

La separación de intereses bidimensional es la más popular, tomada en gran parte del lenguaje Aspect-J; consiste en una especificación asimétrica, en donde existen dos modelos: base y aspectual. El modelo base es aquel que contiene los requerimientos que se pueden encapsular completamente en una sola entidad. El enfoque aspectual es el que contiene los requerimientos transversales y se especifica en función del modelo base. Una segunda perspectiva de trabajo es la multidimensional (Tarr y Sutton, 1999), que consiste en una especificación simétrica, en donde los intereses se especifican de manera homogénea en un solo modelo sin importar si son transversales o no. Las relaciones de impacto entre los intereses se escriben separadamente, manteniendo a los intereses aislados sin relaciones entre ellos. Una comparación gráfica entre ambos paradigmas se muestra en la figura 2.

Aunque el modelo asimétrico o bidimensional ha sido el más utilizado, presenta varios problemas, principalmente derivados de que un solo criterio de descomposición, guía el proceso completo: el desarrollador separa su especificación desde el principio en un modelo base y uno aspectual, sin embargo, él debe decidir cuáles intereses forman parte de cada modelo; realizar cambios en esta decisión posteriormente, resulta en fuertes conflictos adicionales. Esto produce un grado reducido de flexibilidad porque existe dependencia entre ambos modelos: los aspectos deben conocer detalles de los intereses base para poder influenciarlos y, en ocasiones, se obtienen aspectos tan específicos para un escenario que resulta difícil lograr su reutilización.

Por otra parte, en una especificación asimétrica, los requerimientos del modelo aspectual no interactúan entre sí, ni tampoco los del modelo base; esto restringe las relaciones posibles de un requerimiento con otros, lo cual pudiera resultar limitante cuando surgen nuevas necesidades.

UN ENFOQUE ASPECTUAL PARA LA ESPECIFICACIÓN DE REQUERIMIENTOS FORMALES Y REUTILIZABLES

Con la llegada de los aspectos tempranos han surgido diferentes propuestas de trabajo para la especificación de requerimientos aspectuales, ya sean basadas en el modelo bidimensional o en el multidimensional, y en el lenguaje natural o en algún lenguaje matemático formal. Un ejemplo particular de estos enfoques es Z-Siimd (lenguaje Z con separación de intereses con influencia multidimensional) (Roque Hernández, 2011), un enfoque para la especificación de requerimientos usando el lenguaje formal Z donde es posible reutilizar los elementos de las especificaciones realizadas.

El elemento de trabajo de Z-Siimd se muestra en la figura 3. Z-Siimd no exige una organización inicial única de requerimientos dividida en base y aspectual: todos los requerimientos se pueden especificar de manera homogénea con sus estados, entradas y salidas; luego, de manera opcional se organizan por similitud funcional, posteriormente, mediante reglas separadas se establecen las relaciones entre los intereses con sus respectivos detalles de afectación. Las reglas contienen las directrices que un proceso de composición toma para construir un proyecto integrado completo con base en los requerimientos especificados y a las reglas definidas.

Las reglas permiten organizar mejor las interacciones entre los intereses y hacen que su análisis resulte más sencillo; también facilitan el mantenimiento general de las especificaciones al mismo tiempo que dejan sin cambio la estructura original de los intereses, aun cuando las relaciones entre ellos se modifiquen.

El enfoque Z-Siimd considera un repositorio para almacenar requerimientos, intereses y reglas de composición con el objetivo de proporcionar un medio de reutilización. De esta manera, los elementos de un proyecto pueden ser agregados al repositorio, o de él pueden ser extraídos algunos elementos existentes para incorporarlos al diseño de cualquier proyecto en curso. Para facilitar su clasificación y búsqueda, se proporcionan palabras clave o definiciones en lenguaje natural para cada elemento en el proyecto.

CONCLUSIONES

La especificación de requerimientos es una actividad que debe llevarse a cabo en desarrollos de *software* serios con un enfoque de ingeniería. Esta actividad permite establecer metas y evaluaciones objetivas para el proyecto en cuestión, sin embargo, puede resultar compleja debido, entre otros factores, al manejo de los intereses y su separación. El paradigma predominante actualmente, la orientación a objetos, resulta insuficiente para el manejo de los intereses transversales: aquellos que afectan varias partes del sistema, ya que produce *software* que resulta difícil de entender y mantener. La orientación a aspectos es un paradigma relativamente reciente que propone una alternativa de solución a los problemas de la orientación a objetos; surgió en la fase de implementación, pero se ha propagado hasta las primeras etapas de desarrollo como lo son el análisis y el diseño; esta influencia ha producido el término "aspectos tempranos" para referirse a los intereses transversales que aparecen en las etapas previas a la programación. Han surgido diversos enfoques para la captura de requerimientos orientados a aspectos que consideran, ya sea un modelo bidimensional o uno multidimensional para separar los intereses del *software*. Un ejemplo de estos enfoques es Z-Siimd, que propone un modelo de especificación de requerimientos con separación multidimensional de intereses en el lenguaje Z y considera medios de reutilización.

La comunidad de simpatizantes de los enfoques orientados a aspectos ha crecido mucho en los últimos tiempos, principalmente en la etapa de programación con el modelo bidimensional. En la actualidad, la orientación a aspectos es un paradigma que continúa evolucionando y no tiene completamente definidos ni formalizados sus límites y posibilidades. Mientras tanto, el ambiente empresarial aún no lo adopta con seriedad; tal vez por carecer de técnicas estandarizadas adecuadas que soporten las actividades del ciclo de vida completo y garanticen una buena productividad en proyectos a gran escala. Por otra parte, la orientación a aspectos ha capturado la atención de los investigadores y se ha ubicado como un tema de actualidad en la comunidad académica, principalmente con la búsqueda de propuestas para incorporar una adecuada separación aspectual de intereses en las primeras actividades de desarrollo de programas de computadora.

Fuente de elaboración: Ramón Ventura Roque.

FIGURA 3

Modelo de trabajo de Z-Siimd.
Figure 3. Z-SIIMD work model.

REFERENCIAS

Algorry, M. (2005). "Programación orientada a aspectos", en *Revista Users. Code*. 1(3): 60-64.

Baniassad, E., Clements, P., Araujo, J., Moreira, A., Rashid, A. y Tekinerdogan, B. (2006). "Discovering Early Aspects", en *IEEE Software*. 23(1): 61-70.

Braude. (2008). *Ingeniería de software: una perspectiva orientada a objetos*. México: AlfaOmega.

Jacky, J. (2008). *The way of Z practical programming with formal methods*. New York: Cambridge.

Laddad, R. (2003). *AspectJ in action Practical Aspect-Oriented Programming*. Estados Unidos: Manning Publications Co.

Mathiassen, L. y Tuunanen, T. (2011). "Managing Requirements Risks in IT Projects", en *IT Professional*. 13(6): 40-46.

Roque Hernández, R. V. (2011). Tesis doctoral: "Separación multidimensional de aspectos para la especificación y reutilización de requisitos en lenguaje Z". Vigo, Pontevedra, España.

Tarr, P. y Sutton, J. S. (1999). "N Degrees of Separation: Multi-Dimensional Separation of Concerns", en The 21st. Conference on Software Engineering. Los Angeles, California.