Utilización de Casos de Uso durante el Proceso de Desarrollo de Software

María de los Ángeles Fernández Benassati y Elsa Estévez
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Av. Alem 1253 – (8000) Bahía Blanca - Argentina
mdominel@uns.edu.ar
ece@cs.uns.edu.ar

Resumen

La recolección de requerimientos constituye uno de los principales problemas durante el desarrollo de un sistema. Esto se debe a la naturaleza vaga y poco intuitiva de los mismos para los ingenieros de software. La especificación de requerimientos utilizando casos de uso constituye una herramienta que soluciona problemas de comunicación usualmente asociados con la tarea de captura de requerimientos. Si bien los casos de usos se construyen de una manera intuitiva, no existe abundante bibliografía de cómo hacer un buen caso de uso. Generalmente, los analistas se preguntan cuanto deben iterar o cuan detallado debe ser un caso de uso. Habitualmente, no se obtienen respuestas únicas a estas cuestiones y más grave aún, las mismas varían de sistema en sistema. En este trabajo presentamos nuestra línea de investigación basada en la utilización de casos de uso durante distintas etapas del proceso de desarrollo.

1. Introducción

Una de las definiciones de requerimiento propuesta por Shari Pfleeger [1] expresa que un requerimiento es una característica, o una descripción de algo, que el sistema es capaz de hacer con el objeto de satisfacer su propósito. Por esto, un requerimiento es algo que un sistema de software debe brindar a sus usuarios. A modo de ejemplo, podríamos mencionar una función específica, una característica, o un principio que el sistema deba proveer para justificar su existencia. De este modo, los requerimientos definen el alcance del desarrollo de un proyecto de software. Agregar requerimientos implica incrementar el alcance del sistema. Asimismo, los requerimientos estipulan cómo el sistema debería responder a la interacción con el usuario.

Usualmente, los requerimientos son abstractos e intangibles, especialmente para los desarrolladores de software, y es común que se confundan con las soluciones a implementar en el diseño. Sin embargo es crucial mantenerlos separados. De esta manera, los requerimientos se enfocan a *qué hacer* en lugar de *cómo hacerlo*.

Los requerimientos de un sistema pueden ser *funcionales* o *no funcionales.* Los primeros, definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Son aquellos requerimientos que los usuarios necesitan para que el sistema trabaje, y describen una interacción entre el sistema y su ambiente. Los *segundos* son en general globales y bastantes vagos pero hacen al éxito del sistema. Describen características que el sistema debe poseer, y que en caso de no satisfacerlas pueden limitar el uso del mismo. A modo de ejemplo podemos mencionar la performance, las interfaces de usuario, la robustez del sistema, la disponibilidad, la facilidad de mantenimiento, la portabilidad, entre otros.

La actividad de captura y especificación de requerimientos es crucial en toda metodología de desarrollo. Kulak y Guinev [2] enuncian que como primera actividad del ciclo de vida de un sistema, la captura y especificación de requerimientos arma la estructura para el resto de las tareas. De este modo, si la especificación de requerimientos es pobre o incompleta causa que el resto de las etapas, como el análisis, diseño y construcción, estén apoyadas sobre una base débil. Si bien una especificación apropiada y completa no asegura una implementación exitosa, es condición necesaria para que ésta sea posible.

Existen muchas técnicas importantes para escribir especificaciones de requerimientos funcionales, y se clasifican de acuerdo a los diferentes estilos de especificación. Ghezzi, Jazayeri y Mandrioli [3] las clasifican de acuerdo a dos criterios ortogonales diferentes. Las especificaciones se pueden enunciar *formalmente* o *informalmente*. Las especificaciones informales están escritas en lenguaje natural, y pueden usar figuras, tablas y otras notaciones que faciliten la comprensión. Pueden estar estructuradas de una manera estandarizada. Cuando la notación usa una sintaxis y una semántica precisa se convierte en un formalismo. En tal caso, se habla de *especificaciones formales.* Asimismo, es conveniente hablar de *especificaciones semiformales*, ya que muchas veces, en la práctica, se usa una notación concreta pero no se enfatiza una semántica completamente precisa. El diagrama de flujo de datos, usado para la descomposición funcional de un sistema, es un ejemplo de una especificación semiformal, ya que integra la descripción sintáctica formal de las funciones del sistema, y un enunciado informal de su significado.

La segunda distinción importante en los estilos de especificación es entre especificaciones *operacionales* y *descriptivas*. Las *especificaciones operacionales* definen el sistema pretendido describiendo el comportamiento deseado, generalmente proveyendo un modelo del sistema, por ejemplo, un dispositivo abstracto que de alguna manera puede simular su comportamiento. En contraste, las *especificaciones descriptivas* tratan de enunciar las propiedades deseables de un sistema de manera puramente declarativa. Así, sólo enuncian de manera abstracta las propiedades que se deben cumplir, sin proveer ningún algoritmo ni estructura de implementación.

Existen varias técnicas que se pueden aplicar si se desea utilizar especificaciones operacionales. En particular, se pueden mencionar las más ampliamente usadas, tales como Diagrama de Flujo de Datos, Máquina de Estados Finitos, y Casos de Uso. Estos últimos constituyen una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor [4]. Es decir, los casos de usos son una serie de interacciones entre un

sistema y alguien o algo que usa alguno de sus servicios. Como es sabido, todo sistema de software ofrece a su entorno una serie de servicios. En este contexto, un caso de uso es una forma de expresar cómo alguien o algo externo a un sistema usa uno de estos servicios. Se hace referencia a que el servicio puede ser usado por alguien o algo ya que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware y, o, software [11] .

En la próxima sección se presenta la utilización de los casos de uso durante las distintas etapas del proceso de desarrollo.

2. Utilización de Casos de Uso durante el Proceso de Desarrollo

Un caso de uso es un documento narrativo que describe la secuencia de eventos cuando uno o más actores utilizan un sistema para llevar a cabo un proceso. Debido a esto, la identificación de los casos de uso es una de las técnicas más convenientes para la determinación de requerimientos. La principal ventaja de su utilización es que particionan al sistema en un conjunto de piezas lógicas mínimamente relacionadas. Cada una de estas piezas describe una o varias de las funciones que realizará el sistema. Entre otras ventajas que ofrece la utilización de los casos de uso, podemos mencionar:

- *Constituyen un efectivo medio de comunicación*: colaboran en la comunicación del usuario con el equipo de desarrollo porque reflejan la interacción del usuario con el sistema, sin incluir lenguaje técnico o diagramas complicados.
- *Sirven para especificar los requerimientos funcionales y no funcionales*: permiten especificar los requerimientos no funcionales asociados a cada caso de uso utilizando estereotipos o notas.
- *Ayudan a asegurar la trazabilidad de los requerimientos*: proveen la trazabilidad de los requerimientos a través de las distintas etapas del ciclo de vida, ya que los casos de uso representan bloques de construcción para el diseño, implementación, testeo y entregas del sistema.
- *Desalientan diseños prematuros*: sirven para detectar aquellas situaciones en las cuales el diseño se mezcla con la especificación de requerimientos.

Los casos de uso intervienen en la mayoría de las etapas de desarrollo de un sistema. En la planificación del proyecto se pueden usar como base para la estimación del esfuerzo requerido [6, 7, 8, 9]. En el modelado del negocio es posible utilizarlos para la documentación de la reingeniería de procesos [5]. En la etapa de análisis, los diagramas de secuencia utilizados para modelar el comportamiento, se corresponden directamente con los casos de uso. Durante la actividad de diseño, los casos de uso se transforman agregándoles consideraciones físicas especificando detalles de implementación. El diagrama de casos de uso y su jerarquía puede ser útil en el diseño de la interfaz del usuario ya que muestra como los usuarios parten lógicamente su sistema en abstracciones. La jerarquía de los casos de uso contribuye en el mecanismo de navegación del sistema ya sea por medio de menús, botones o páginas web. Asimismo es posible utilizar los casos de uso para definir las políticas de seguridad del sistema.

En el ciclo de vida evolutivo, es posible utilizar los casos de uso para decidir qué funcionalidad incluir en cada incremento[12]. En otros ciclos de vida, los casos de uso se pueden utilizar como los bloques de construcción durante la etapa de implementación. En el testing, los casos de testeo son descripciones detalladas de las interacciones entre los actores y el sistema. Debido a esto, es muy útil mapear los casos de uso con los casos y escenarios de testeo. Este método provee una excelente posibilidad de seguir los requerimientos originales a ser testeados en el sistema. Durante la etapa de mantenimiento, se pueden utilizar los casos de uso para decidir qué funcionalidad estará involucrada en cada versión de mantenimiento. La decisión de qué funcionalidad se libera con cada versión es importante, ya que de esta manera el usuario se notifica de qué parte del sistema es entregada. Por último, la utilización de los casos de uso posibilita dimensionar el impacto de los cambios que se suceden durante las etapas del desarrollo, y esto resulta de utilidad para el correcto gerenciamiento del proyecto.

3. Conclusiones y Trabajo Futuro

Las técnicas de especificación de requerimientos funcionales tradicionales, como el diagrama de flujo de datos, el modelo de entidad relación, o los prototipos son útiles en momentos muy específicos del desarrollo de un sistema. Por ejemplo, el modelo de entidad relación es útil para el modelado de la estructura de datos, para el diseño de la base de datos, y al comienzo de la implementación. El diagrama de flujo de datos es útil para la etapa de modelamiento de las funciones, pero no sirve para el resto de las tareas de desarrollo. Por otra parte, los prototipos son bien recibidos por los usuarios, pero los alienta a concentrarse en los detalles de la interfase y no en los requerimientos del sistema. A diferencia de estas, los casos de uso pueden utilizarse a lo largo de la mayoría de las actividades del desarrollo, como se mencionó anteriormente. En base a esto, constituyen una herramienta invalorable para las distintas etapas del proceso de desarrollo.

En una primera etapa nuestros trabajos están dirigidos a investigar la relación existente entre los casos de uso y la técnica de puntos de función para cálculo del esfuerzo de desarrollo. Los puntos de función es uno de los métodos más difundidos para medir aplicaciones de software desde la perspectiva de los requerimientos[10]. Nuestro trabajo se centrará en la elaboración de métricas a partir de la combinación de la técnica de puntos de función y casos de uso. En las etapas posteriores, extenderemos nuestras investigaciones a fin de lograr un enfoque metodológico para la utilización de los casos de uso en cada una de las etapas, incluyendo el análisis de riesgo, el diseño de la arquitectura, el testing entre otras. Complementariamente, analizaremos la posibilidad de la inclusión formal de la especificación de requerimientos no funcionales en los casos de uso para dominios de aplicación específicos, como por ejemplo aplicaciones de web, y modelamiento de procesos de negocio.

4. Bibliografía

[1] Shari Lawrence Pfleeger. Ingeniería de Software, Teoría y Práctica. Prentice Hall, 2002.

[2] Daryl Kulak, Eamonn Guiney. Use Case Requerements in Context, Addison Wesley, 2002.

[3] Ghezzi, Jazayeri, Mandrioli. Fundamentals of Software Engineering, Prentice Hall, 1991.

[4] Booch, Rumbaugh, Jacobson . El Lenguaje Unificado de Modelado. Addison Wesley, 2000.

[5] Jacobson, Ericson. Object Advantage, The Business Process Reengineering with Object Technology. Addison Wesley, 1995.

[6] Anda Bente. Comparing Effort Estimates Based on Use Case Points with Experts Estimates. Empirical Assessment in Software Engineering, (EASE 2002).

[7] Estimating Software Develoment Effort Based on Use Cases – Experiences from Industry. Lecture Notes for Computer Science 2185, págs. 487-502, 2001.

[8] Dekkers, C. Function Point and Use Cases – Where´s the Fit?. http://www.quialityplustech.com/FPUUSeCases_files/frame.htm.

[9] Longstreet, D. Use Cases and Function Points. http://www.softwaremetrics.com/Articles/usecases.htm

[10] International Function Points Users Group http://www.ifpug.org

[11] Cokburn, A. Writing Effective Use Cases, Addison-Wesley, 2000.

[12] Bittner, K. Driving Iterative Development with Uses Cases. IBM, 2004 http://www-106.ibm.com/developerworks/rational/library/4029.html.

[13] Plasil F., Mencl V. Getting “Whole Picture” Behavior in a Use Case Model. Dept.of Computer Science, University of New Hampshire, TR 02/11, Durham 2002.

[14] Hausmann J., Hecke R., Taentezer. Detection of Conflicting Functional Requirements in a Use Case Driven Approach, ICSE, 2002, Orlando, Florida.