Extensión y Aplicación de AspectZ a la Administración de un Sistema de Fichas de Salud Electrónicas en Chile

Cristian Vidal⁽¹⁾, Carolina Del Río⁽²⁾ y Rodrigo Saens⁽²⁾

(1) Escuela de Ingeniería Informática Empresarial, Facultad de Ciencias Empresariales, Universidad de Talca Campus Lircay, Avenida Lircay s/n, Talca-Chile (e-mail: cvidal@utalca.cl).

(2) Escuela de Ingeniería Comercial, Facultad de Ciencias Empresariales, Universidad de Talca Campus Lircay, Avenida Lircay s/n, Talca-Chile (e-mail: cdelrio@utalca.cl; rsaens@utalca.cl).

Recibido Feb. 28, 2012; Aceptado Mar. 29, 2012; Versión final recibida May. 10, 2012

RESUMEN

Este artículo presenta una extensión de AspectZ con el objetivo de modelar los requerimientos del software que administra un sistema de fichas electrónicas de salud en Chile: SYSFIC 1.0. AspectZ es un lenguaje de especificación formal orientado a aspectos que preserva las características bases del lenguaje Z y que ayuda a encapsular incumbencias transversales del sistema en entidades separadas e independientes. La aplicación incluye el uso de diagramas con Lenguaje Unificado de Modelado (LUM) y un modelado formal del sistema con AspectZ. Se discuten, además, las principales características del paradigma de modelado orientado a aspectos, como la separación incumbencias y la solución de los problemas del código disperso y enmarañado. Los resultados de este trabajo muestran que la unión del modelado orientado a aspectos y de un método formal de desarrollo de software permite obtener soluciones más confiables para sistemas de información críticos como SYSFIC.

Palabras clave: AspectZ, modelación orientada a aspectos, lenguaje unificado de modelado, SYSFIC

Extension and Application of AspectZ to the Management of an Electronic Health Card System in Chile

ABSTRACT

This paper presents an extension of AspectZ to model the requirements of software that manages a system of electronic health records in Chile: SYSFIC 1.0. AspectZ is an aspect-oriented language that preserves the basic characteristics of language Z and that aims to encapsulate cross-cutting concerns of a system into separate and independent entities. The application includes the use of Unified Modeling Language (UML) diagrams and a formal modeling of the system using AspectZ. The main features of the aspect-oriented modeling paradigm, such as the separation of concerns and the reduction and elimination of scattered and tangled code, are discussed. The results of this study show that the combination of aspect-oriented modeling and a formal method of software development allow obtaining more reliable solutions for critical information systems such as SYSFIC.

Keywords: AspectZ, aspect-oriented modeling, unified modeling language, SYSFIC

INTRODUCCIÓN

La orientación a aspectos (OA) representa un nuevo y promisorio paradigma de programación. Basándose en el principio de separación de intereses o incumbencias, la OA define una moderna forma de trabajo que modulariza en entidades independientes las diversas funcionalidades que conforman una aplicación, solucionando dificultades que a la tradicional orientación a objetos le son muy difíciles de resolver. De hecho, son precisamente esos problemas complementarios de código disperso (*scattered code* o *scattering*) y código enmarañado (*tangled code* o *tangling*) los que, según Jacobson *et al.* (2004), impiden que las implementaciones resulten más amables y comprensibles.

De acuerdo a Pressman (2001), el desarrollo de software orientado a aspectos (DSOA), al ser un nuevo paradigma de programación formal, requiere prestar atención a todas y cada una de las etapas del ciclo de desarrollo; a saber: especificación y análisis de requerimientos, diseño, implementación y mantención del software.

Aunque desde sus inicios DSOA permite una ventaja sustancial en las fases de implementación y mantención —las dos últimas etapas del ciclo de desarrollo del software—, hasta el momento sólo existe una escasa experiencia en modelación formal orientada a aspectos que permita evaluar la utilización del lenguaje de especificación formal AspectZ.

Siguiendo los trabajos de Skipper (2004), Jacobson *et al.* (2004), Yu *et al.* (2005), Li (2009) y Vidal *et al.* (2012), este artículo presenta una extensión de AspectZ, cuyo objetivo es modelar los requerimientos de software aplicado a la gestión de un sistema de fichas electrónicas de salud en Chile, SYSFIC 1.0. La aplicación incluye el uso de diagramas de casos con lenguaje de modelamiento unificado (UML, por su sigla en inglés) y una modelación formal del sistema con la utilización de AspectZ.

Este artículo se estructura de la siguiente forma. En la sección que sigue se describe el paradigma de DSOA junto a los principales inconvenientes no resueltos por paradigmas previos de desarrollo de software. A continuación, secciones con la descripción de los diagramas de casos de uso UML orientado a aspectos y el lenguaje de especificación formal AspectZ. Luego, una sección que presenta el caso de estudio junto a los diagramas de casos de uso UML orientados a aspectos asociados. Así, la penúltima sección presenta la modelación formal de SYSFIC usando AspectZ, detallando extensiones propuestas para una modelación efectiva. Finalmente, la última sección detalla las principales conclusiones de este artículo y las posibilidades de implementar SYSFIC con Aspect/J.

DESARROLLO DE SOFTWARE ORIENTADO A ASPECTOS

Aunque de acuerdo a Chávez *et al.* (2006) y Barnes (2007) la programación orientada a objetos permite una mejor modularización que la programación estructurada —con lo que se facilita la escritura, depuración y mantenimiento del código de aplicación—, la existencia de problemas complementarios, como el código disperso y el enmarañado, lleva a que la orientación a aspectos siga siendo hoy un tema recurrente de investigación para el desarrollo de software.

Si bien la principal meta del DSOA es lograr una completa separación de incumbencias, Jacobson *et al.* (2004) y Vidal *et al.* (2012) explican que gracias a la eliminación del código enmarañado y disperso, es posible trabajar sin incumbencias cruzadas no sólo en la etapa de programación, sino durante todo el desarrollo del software, con el objetivo de lograr una mejor abstracción modular del sistema.

CÓDIGO DISPERSO (SCATTERING CONCERNS)

En el desarrollo de software es común encontrar elementos funcionales o no funcionales que están presentes en más de un módulo o componente de la aplicación. Esto, que a simple vista no

representa un mayor inconveniente, sí se transforma en un problema complejo cuando los requisitos o el diseño de alguno de estos elementos cambian, haciendo necesario realizar actualizaciones en la aplicación en cada lugar donde dicho elemento se localiza. Este fenómeno de expansión es conocido como código disperso. De acuerdo a Jacobson *et al.* (2004), el código disperso dificulta la comprensión durante el desarrollo de un sistema de software, lo que se traduce en un mal mantenimiento que complica la realización de mejoras, particularmente en sistemas con software de gran tamaño. La figura 1, obtenida de Vidal *et al.* (2011), muestra una funcionalidad que se expande y aparece junto a otras funcionalidades en el desarrollo de una aplicación bancaria web.

Fig. 1: Ejemplo de código disperso

CÓDIGO ENMARAÑADO (TANGLING CONCERNS)

Se denomina enmarañado al problema que se produce cuando un mismo módulo es utilizado para satisfacer distintas necesidades del sistema. Existe un elemento base, y adicionalmente uno o más elementos que se preocupan de otras situaciones anexas para el logro o total cumplimiento de la idea central. En otras palabras, es común que un módulo de un sistema de *software*, en lugar de dedicarse sólo a una tarea en particular, deba realizar o respetar más de una tarea, restando énfasis a su funcionalidad o tarea principal. Si un cambio en alguno de estos elementos adicionales fuera necesario, deberá realizarse cambios en variados elementos de la aplicación, con las consecuentes dificultades en el mantenimiento y desarrollo del software del sistema. De esta forma, si fuera posible lograr una completa separación de tareas, donde una tarea base no se preocupara de la existencia de tareas anexas, como explican Jacobson *et al.* (2004), definitivamente se obtendría un mejor desarrollo de software. La figura 2, obtenida de Vidal *et al.* (2011), presenta un ejemplo de enmarañado o mezcla de funcionalidades en el desarrollo de una aplicación bancaria web.

Al resolver los problemas de código disperso y enmarañado, la orientación a aspectos permite una completa modularización, lo que se traduce en elementos más limpios y fáciles de entender, obteniendo como resultado un software más fácil de mantener, extender y reutilizar. La figura 3 ilustra el efecto de aplicar DSOA sobre paradigmas tradicionales de desarrollo de software que producen un alto grado de código disperso y enmarañado (o mezcla) de funcionalidades.

Fig. 2: Ejemplo de enmarañado

Fig. 3: Aplicación de DSOA

CASOS DE USO DE UML ORIENTADO A ASPECTOS

Un diagrama de caso en que se aplique UML orientado a aspectos reúne todas las características originales de diagramas de casos de uso UML, pero con la reformulación semántica de la relación de extensión, la que, en lugar de reemplazar por completo el caso de uso extendido, representa la incorporación de nuevo comportamiento en situaciones particulares (aspectos). La figura 4, obtenida de Vidal *et al.* (2012), presenta un ejemplo de modelación orientada a aspectos con diagramas de casos de uso UML de un sistema informático con uso de *Login*. Como se aprecia en esta figura la relación de extensión en Diagramas de Casos de Uso es un esquema útil para la modelación de incumbencias cruzadas.

Fig. 4: Diagrama de Casos de Uso de Sistema con Uso de Login.

Es importante recalcar que además del cambio semántico de la relación de extensión en la modelación gráfica de casos de uso UML, hay también cambios relativos a su modelación en su versión original. Como explican Jacobson *et al.* (2004) y Vidal *et al.* (2012), en la especificación textual de casos de uso orientados a aspectos se declaran puntos de extensión en los elementos base y puntos de corte de extensión en los aspectos, como "Sistema" y "Login" en figura 4.

LENGUAJE FORMAL ASPECTZ

Como señalan Bowen (1996) y Vidal *et al.* (2011), Z es un lenguaje de especificación formal basado en la teoría de conjuntos que permite la representación matemática de datos y operaciones de un sistema de *software*. Respecto a la definición de información, Z soporta la definición de constantes, tipos básicos y tipos compuestos de datos. Por sintaxis, entre paréntesis cuadrado se define un tipo básico, mientras que para definir un tipo compuesto se escribe el nombre del tipo que se asigna a la composición de tipos y/o constantes. La definición de constantes es similar a la definición de tipos compuestos.

De acuerdo a Yu *et al.* (2005), AspectZ es una extensión al lenguaje formal Z con la inclusión de conceptos del paradigma DSOA, gracias a lo cual es posible una separación de elementos bases y cruzados del sistema de manera independiente. AspectZ es un lenguaje de especificación formal orientado a aspectos que preserva las características bases de lenguaje Z, donde el principal elemento de modelación es el esquema. AspectZ preserva las ideas de esquema base y esquema inicial y, gracias a la inclusión de esquemas de aspectos, habilita la obtención de una separación de incumbencias o asuntos en los esquemas de operación. En este caso, un esquema de operación está encargado sólo de su función principal, mientras que los detalles adicionales o situaciones especiales que podrían ocurrir sobre la operación son considerados en esquemas de aspectos. De esta forma, como señalan Yu *et al.* (2005), AspectZ permite una completa modularización en la etapa de análisis de requerimientos gracias a la temprana detección de aspectos.

Algunos elementos de AspectZ son la identificación de una referencia a un esquema con símbolo Ω junto a la declaración de un punto de corte que permite identificar un punto de unión entre el esquema de aspectos y el esquema referenciado (lugar donde un aspecto debe aparecer automáticamente), además de las palabras 'Insert' y 'Replace' para indicar si el comportamiento o reglas son agregadas antes de la operación o para modificar ciertos elementos ya existentes en la misma. De esta forma, AspectZ incluye los principales elementos de DSOA, donde cada módulo o funcionalidad base no se preocupa explícitamente de la aparición de las incumbencias cruzadas, ya que son ellas quienes deben indicar el punto de corte asociado al punto de unión para su aparición. La figura 5, obtenida de Vidal *et al.* (2012), muestra la estructura y los componentes de un esquema AspectZ.

Fig. 5: Esquema AspectZ y sus componentes.

En la sección siguiente se presentan los requerimientos de SYSFIC 1.0, software que administra un sistema de fichas electrónicas de salud en Chile, y su diagrama de casos de uso UML orientado a aspectos asociado.

CASO DE ESTUDIO Y APLICACIÓN DE UML ORIENTADO A ASPECTOS

SYSFIC es un sistema de fichas electrónicas de pacientes que busca lograr una atención más eficiente en los diferentes servicios de salud en Chile. Según Muñoz y Pérez (2011), los requerimientos textuales de SYSFIC son una ficha electrónica en línea, donde se registre toda la hoja de vida de cada paciente para que sea abierta en cualquier nodo de la red asistencial de servicios de salud en Chile (hospitales, clínicas, consultorios generales urbanos y rurales, postas rurales de salud y estaciones médico rurales). La ficha electrónica debería llevar toda la información necesaria para la atención adecuada del paciente. Por ejemplo, sus datos personales y sus datos clínicos, así como desde qué parte de la red nacional ha sido derivado, las patologías por las cuales ha consultado, las fechas de estas consultas, los doctores y especialistas que lo han atendido, los medicamentos que se le han recetado, antecedentes familiares de enfermedades hereditarias, diagnósticos efectuados por doctores y fechas, entre otras informaciones.

Si el Servicio Nacional de Salud de Chile tuviera esta información en red, las colas de espera de los hospitales y los tiempos de atención a los pacientes, por ejemplo, disminuirían; los diagnósticos serían más certeros y el paciente incurriría en menos gastos —ya que no necesitaría ir a doctores que ya lo hubiesen atendido antes para pedir su ficha clínica—; obligando además a los médicos a ser más precisos en sus diagnósticos y tratamientos. El mismo organismo público tendría también valiosa información que le permitiría planificar con mayor exactitud la situación sanitaria de la población, los especialistas, medicamentos y equipos médicos más requeridos por la gente, etc.

De acuerdo a esta especificación de requerimientos de SYSFIG, la figura 6 presenta la modelación gráfica usando diagramas de casos de uso orientados a aspectos.

Fig. 6: Diagrama de casos de uso orientado a aspectos en SYSFIG.

La figura 7 muestra la especificación textual del caso de uso Autenticación que representa un aspecto. En esta descripción textual, se indican los puntos de corte de extensión en los cuales el caso de uso Autenticación debe aparecer. En este ejemplo, el punto de extensión es declarado en el caso de uso Historial.

Es importante considerar que, en iteraciones y revisiones futuras de SYSFIG pueden aparecer nuevos aspectos.

Fig. 7: Diagrama textual de casos de uso autenticación de SYSFIG.

En la sección siguiente se desarrolla una extensión de AspectZ cuyo objetivo es modelar los requerimientos del software que administra un sistema de fichas electrónicas de salud en Chile: SYSFIG 1.0.

CASO DE ESTUDIO, APLICACIÓN Y EXTENSIONES DE ASPECTZ

A continuación, se presenta la modelación formal con AspectZ del ingreso, la autenticación y el registro de pacientes y sus enfermedades en SYSFIG. La figura 8 muestra la declaración de tipos básicos y compuestos para esta versión reducida de SYSFIG, mientras la figura 9 representa los esquemas de espacio de estados e inicial de SYSFIG.

Fig. 8: Tipos Básicos y Compuestos de Especificación AspectZ de SYSFIG.

Nótese que en la figura 8 la inclusión de Mensaje y Tipo son parte de la modelación de aspectos. Asimismo, en la figura 9 se aprecian 2 invariantes respecto a las relaciones existentes entre conjuntos de pacientes y diferentes enfermedades cuando un paciente registra una determinada patología.

Fig. 9: Esquema Z tradicional SYSFIG para la Inserción de Información.

Asumiendo que Tipo sólo considera los valores 1, 2 y 3 para el ingreso, enfermedades asociadas y datos médicos de cada paciente, en la figura 9 se aprecia el esquema INSERTAR en Z tradicional, mientras en la figura 10 su modelación AspectZ para que, dado el tipo de ingreso, se proceda al ingreso de pacientes.

Fig. 10: Esquema AspectZ SYSFIG para la Inserción de Información - Pacientes.

Note que se asume que T? representa el valor de entrada de Tipo, donde en este caso corresponde al valor 1. En AspectZ, para realizar esta validación se utiliza un Aspecto Aspect1 y el esquema INSERTAR se encarga de que el usuario sólo indique lo que desea ingresar que

representa su función principal. De esta forma, son necesarios esquemas de aspectos para verificar los otros posibles valores de T?. Estos esquemas son similares a Aspect1.

Como se muestra en las figuras 8 y 9, el enfoque de modelación orientada a aspectos implica una modelación adicional por cada situación particular que se considere aspecto, aunque el tamaño del sistema —en modelación y codificación— sería similar al que se obtiene utilizando los paradigmas tradicionales. No obstante, es útil para demostrar la aplicabilidad de AspectZ y modelación formal orientada a aspectos.

En modelación Z y AspectZ, para cada esquema de operación de estos lenguajes existe una posibilidad de éxito o error cuando no se satisface alguna precondition. De esta forma, para cada operación es posible incorporar un aspecto que sólo aparece al finalizar la operación. Con esta finalidad, se propone la inclusión de la palabra 'After', similar a 'Insert' y 'Replace', además de 'FINISH' para conocer si la operación finalizó correctamente (si satisface todas las precondiciones), o 'NOT_FINISH' en caso contrario. Además, es necesario introducir el símbolo de composición '+' para indicar composición al final de la operación. De esta forma, para un aspecto que debiera aparecer al final de una operación, no es necesario señalar explícitamente un punto de corte, como se aprecia en la Figura 11. Nótese que gracias a este esquema de aspecto, el esquema Aspect1 no requiere declarar ni trabajar sobre M!.

Fig. 11: Esquemas Aspect Mensaje AspectZ y Esquema de Integración.

CONCLUSIONES

DSOA es un nuevo paradigma en la ingeniería de software que entrega la posibilidad de compartimentar en módulos separados las diversas propiedades o aspectos que intervienen en la aplicación final del desarrollo de software, como así también de obtener, a través de su modelación formal, una comprensión más profunda de todos y cada uno de los aspectos que interactúan en un sistema.

Dado que los diagramas de casos con UML son necesarios para la obtención y análisis de requerimientos, y puesto que los métodos formales son imprescindibles para el desarrollo de software crítico que manipula variables de alto valor, como el software médico, este artículo muestra la aplicación de diagramas de casos de uso orientados a aspectos, además de una modelación formal del sistema con el uso de AspectZ. Ambas herramientas de modelación se aplican al caso particular de un sistema de fichas electrónicas de salud para la red asistencial de servicios de salud en Chile: SYSFIC 1.0. La modelación orientada a aspectos permite lograr una separación de incumbencias y también es consistente con el uso posterior de herramientas de programación orientadas a aspectos, como Aspect/J o Aspect C#.

AspectZ permite la unión de dos visiones: la de DSOA y la de modelación formal. Se logra así un mejor modelamiento para la detección temprana de incumbencias cruzadas y, a su vez, una modularización total del sistema. Este artículo presenta, además, una extensión de AspectZ para una integración al final de una operación, junto con el símbolo '+' para determinar una inclusión considerando dos casos posibles.

Si bien AspectZ fue propuesto hace siete años por Yu et al. (2005), a la fecha aún no se han presentado trabajos que muestren el uso y aceptación práctica de este lenguaje formal. Como trabajo futuro se encuentra la validación de AspectZ en el desarrollo rápido de software, además de establecer empíricamente la real validez de la identificación temprana de incumbencias. Se está actualmente trabajando en la aplicación y extensión de lenguaje formal ObjectZ en DSOA. Desde el punto de vista SYSEFIG, se considera la implementación del sistema en ambiente web con la utilización Aspect/J.

REFERENCIAS

- Barnes, D. J., *Programación Orientada A Objetos Con Java Bluej*, 3ª edición, Prentice Hall (2007).
- Bowen, J., *Formal Specification and Documentation Using Z: A Case Study Approach*, Center for Applied Formal Methods, London South Bank University (1996).
- Chávez, C., García, A., Kulesza, U., Sant Anna, C. y Lucena, C., *Crosscutting Interfaces for Aspect-Oriented Modeling*, Journal of the Brazilian Computer Society 12(1): 43-58 (2006).
- Jacobson, I. y Ng, P., *Aspect Oriented Software Development with Use Cases*, 1ª edición. Addison Wesley Professional, Nueva York, USA (2004).
- Li, G., *Identifying Crosscutting Concerns in Requirement Specifications: A Case Study*, Tesis de Magíster en Ciencias, Queen's University, Kingston, Ontario, Canada (2009).
- Londoño, L.F., Anaya, R. y Tabares, M. S., *Análisis de la Ingeniería de Requisitos Orientada por Aspectos Según la Industria del Software*, Revista EIA-Escuela de Ingeniería de Antioquia 09: 43-52 (2008).
- Muñoz, P. y Pérez, N. *Análisis de Listas de Espera del Hospital Regional de Talca*. Tesis de Titulación. Escuela de Ingeniería Comercial, Universidad de Talca, Chile (2011).
- Pressman, R., *Ingeniería del Software. Un Enfoque Práctico*, 5ª edición, McGraw Hill (2001).
- Skipper, M. C., *Formal Models for Aspect-Oriented Software Development*, Tesis de Doctorado, Colegio Imperial de Londres, Londres, Inglaterra (2004).
- Tabares, M. S., Alférez, G. H. y Alférez, E. M., *El Desarrollo de Software Orientado a Aspectos: Un Caso Práctico para un Sistema de Ayuda en Línea*, Revista Avances en Sistemas e Informática 05(02): 61-68 (2008).
- Vidal, C., Hernández, D., Gutiérrez, C., Meza, R. y López, L., *Modelación Formal Orientada a Aspectos Usando AspectZ*. Encuentro Chileno de Computación 2011, Curicó, Chile (2011).
- Vidal, C. L., Hernández, D. D., Pereira, C. A. y Del Río, M. C., *Aplicación de Modelación Orientada a Aspectos*, Revista Información Tecnológica (CIT), Universidad de La Serena, Chile: 23(1), 3-12 (2012)
- Yu, H., Liu, D., Yang, J. y He, X., *Formal Aspect-Oriented Modeling and Analysis by Aspect-Z*, Actas de 17th International Conference on Software Engineering and Knowledge Engineering, SEKE'2005, 124-132, Taipei, Taiwan, Republic of China, 14 al 15 de Julio (2005).