Estrategias para la Integración y Conexión de Reglas de Negocio con Aspectos

Sandra Casas[1], Claudia Marcos[2], Héctor Reinaga[1], Juan G. Enriquez[1], Graciela Vidal[1] y Franco Herrera[1]
[1]Unidad Académica Río Gallegos, Universidad Nacional de la Patagonia Austral, Río Gallegos, Argentina
[2]ISISTAN, Universidad Nacional del Centro, Tandil, Argentina
lis@uarg.unpa.edu.ar

CONTEXTO

El grupo de investigación ha desarrollado diferentes actividades relacionadas al Desarrollo de Software Orientado a Aspectos desde el año 2005. Estas se podrían clasificar en tres líneas principales: Conflictos entre Aspectos, Aspectos Tempranos y Minería de Aspectos y Refactoring Aspectual. Asimismo se han ejecutado exitosamente tres proyectos de investigación y publicado casi 40 artículos.

RESUMEN

La Programación Orientada a Aspectos (POA) ha sido propuesta como una alternativa para implementar (encapsular) las conexiones entre reglas de negocio (RN) y los componentes funcionales o lógica de negocio (funcionalidad base) con el objeto de minimizar las dependencias y acoplamiento. Trabajos previos demuestran que en consecuencia se logra mayor reutilización del código base y mejoras en el mantenimiento del software. Sin embargo, otros problemas aparecen directamente asociados con la naturaleza de las conexiones y las limitaciones de los lenguajes POA que dificultan el desarrollo del software.
Este proyecto de investigación plantea explorar, proponer y experimentar estrategias que ofrezcan mayor flexibilidad en la implementación de conexiones con aspectos, superando los enfoques existentes. Las bases del trabajo se establecen en las siguientes premisas: a) plantear el abordaje de enfoques POA declarativos para el encapsulamiento de las conexiones entre reglas de negocio y la funcionalidad base; b) definir especificaciones de alto nivel para la representación de las conexiones independientes de lenguajes de programación y notaciones de diseño; c) establecer una taxonomía que clasifique a las conexiones de acuerdo a sus características a partir de las especificaciones para su procesamiento automático.
Los resultados del proyecto de investigación se apoyarán en el desarrollo de prototipos y casos de estudio que permitan demostrar la validez de los conocimientos generados.

Palabras clave: *Programación Orientada a Aspectos, Reglas de Negocio, AspectJ, Spring.*

1. INTRODUCCION

1.1 El problema de las conexiones

Las reglas de negocio (RN) reflejan las restricciones que existen en toda organización, de modo que nunca sea posible llevar a cabo acciones no válidas [3]. Las RN son restricciones de tipo legal, político, competitivas, de oportunidad, etc. Una aplicación software implementa un conjunto de RN para dar respuesta acorde a los requerimientos del cliente. Típicamente una RN está representada por una sentencia "***if*** *condición* ***then*** *acción*", en las aplicaciones Orientadas a Objetos (OO) suele estar encapsulada en una clase. Los mecanismos habituales de implementación de RN requieren embeber la evaluación de la regla o su invocación en los módulos de la funcionalidad base, causando que la implementación se disperse y confunda por múltiples módulos. Un cambio en la especificación de la RN requiere cambios en todos los módulos que están involucrados. Estas modificaciones resultan invasivas y consumen tiempo. Además debido a que las RN son mucho más volátiles que la lógica de negocio, al estar mezcladas causan que la lógica de negocio se vuelva también volátil. Este tipo de implementación de las RN ocasiona inconvenientes para la reutilización y mantenimiento del código.

Con el objeto de minimizar las dependencias se han aplicado técnicas OO para la separación e integración de RN. Particularmente el Patrón Objeto Regla [1] fue creado para representar las RN como objetos. Este patrón puede llegar a ser bastante elaborado, si se aplica conjuntamente con otros patrones de diseño como Composite, Strategy, etc. [13], para minimizar la dependencia y mejorar la reutilización. Sin embargo, estas alternativas tienen por inconveniente que requieren demasiados objetos intermedios, los cuales se tornan complejos de manejar y persisten ciertos problemas de integración de programa: dependencia, adaptación y combinación de reglas. Estos problemas han sido discutidos por [20][14].

1.2 Programación Orientada a Aspectos

La Programación Orientada a Aspectos (POA) [17] es un nuevo paradigma para el desarrollo software que proporciona mecanismos y abstracciones para la implementación de crosscutting concerns (Seguridad, Logging, Autentificación, Persistencia, Concurrencia, etc.), de manera separada y aislada. Es decir, la orientación a aspectos (OA), es una

técnica que permite aplicar el principio de Separación de Concerns [12][15] y de esta forma, lograr una mayor y mejor modularización del código. El enfoque resulta muy prometedor y atrayente ya que supera las características indeseables producidas por el efecto de "tiranía de la descomposición dominante" [23] que tiene como consecuencias negativas la generación de código mezclado y diseminado. La POA provee mecanismos que permiten aplicar una mayor descomposición modular a los sistemas, así los sistemas son más fáciles de diseñar, codificar, mantener y reusar.

Las herramientas POA en general extienden lenguajes de programación convencionales mediante la adición de nuevos constructores sintácticos y semánticos que permiten la descomposición, conocidos como join-points, pointcuts, advises, etc. y una unidad o módulo denominada "aspecto" que agrupa estos constructores y además incluye el código a insertar. Un proceso de tejido (weaver) o composición (realizado por un nuevo tipo de compilador o intérprete) combina los componentes de funcionalidad base con los aspectos, para generar la aplicación ejecutable [21].

Especialmente, a partir de la aparición de AspectJ [18] el paradigma ha recibido especial interés y múltiples aportes provenientes tanto del campo de la investigación como la industria. Desde entónces han aparecido una importante cantidad de lenguajes de programación, herramientas de desarrollo y frameworks que en distinta medida y forma dan soporte al enfoque POA [21]. Los principios de la OA se han trasladado rápidamente a otras temáticas relacionadas con el desarrollo del software como la Ingeniería de Requerimientos [22], Análisis y Diseño Aspectual [5], Técnicas de Refactoring [2], Métricas POA [4], etc.

1.3 Programación Orientada a Aspectos y Reglas de Negocio

La POA ha sido propuesta para mejorar la integración de RN con la funcionalidad base. Los enfoques están dirigidos a encapsular las RN y su conexión en los aspectos o ha encapsular sólo la conexión de la RN. Estos trabajos se han centrado en proveer soluciones con AspectJ y JasCo [24]. Las experiencias indican que aunque se mejora la encapsulación, minimiza la dependencia y favorece la reutilización, aparecen otros inconvenientes (más adelante se detallan). No obstante, los mismos autores indican que es imposible generalizar los resultados a todos los enfoques OA debido a que algunos de ellos son fundamentalmente demasiado diferentes. A continuación se mencionan las contribuciones más importantes relacionadas:

[19] presenta una plantilla para la implementación de RN en AspectJ. De manera general, una RN es encapsulada en un aspecto, en el pointcut se establece los métodos (eventos) en los actuales la RN debe ser aplicada y en los advice se codifica la RN (evaluación de condición y acción). Desde este enfoque una RN es encapsulada en un aspecto junto con su integración (conexión). Este enfoque le da mayor reutilización y estabilidad a la funcionalidad base, ya que separa la RN de la misma; de igual forma, hace más fácil su mantenimiento, pero exhibe un nivel de dependencia importante entre la RN y la funcionalidad base que impactan en el reuso de la misma.

En [16] se presenta una experiencia real de refactorización de RN en aspectos con AspectJ, de una aplicación J2EE importante. La estrategia se basó en la implementación de una jerarquía de aspectos paralela a la jerarquía de los objetos de acceso a los datos.

[6] aporta una serie de contribuciones directamente relacionadas con la problemática de las conexiones entre RN y la funcionalidad base en software OO con aspectos. Brevemente éstos se exponen a continuación:

- Se identifican los requerimientos tecnológicos para lograr soporte para la encapsulación de las conexiones (conectores), independientes de los mecanismos de implementación. Estos son: conectores que dependan de propiedades dinámicas; pasaje de la información necesaria (anticipada, no anticipada, contextual, no contextual); reuso de enlaces a las RN en diferentes eventos; combinación, priorización y exclusión de RN cuando interfieren unas con otras; control de instanciación, inicialización y ejecución de enlaces de RN; cumplimiento dinámico de los requerimientos planteados para no interrumpir la ejecución de los sistemas [10].

- Se analiza el soporte provisto por AspectJ [7] y JasCo [11] en el cumplimiento de los requerimientos identificados. AspectJ permite desacoplar las diferentes partes que constituyen la conexión de las RN en aspectos separados que pueden reutilizarse independientemente. Sin embargo, esto produce una proliferación de aspectos que se tornan difíciles de manejar. Generalmente las relaciones entre los aspectos se expresan en los mismos aspectos, lo que reduce la reusabilidad de aspectos y composición. Además AspectJ tiene algunas características muy poderosas y de bajo nivel que se usan para resolver una amplia gama de problemas. Sin embargo, a veces la misma característica es usada para solucionar semánticamente diferentes concerns, en consecuencia se dificulta la comprensión y portabilidad de programas. Aunque la reutilización del código del aspecto es posible a través de la herencia, los pointcuts de AspectJ son frágiles cuando definen directamente un evento concreto en la ejecución y por consiguiente son menos reusables. Adicionalmente la instanciación e inicialización son controlados por el mismo framework, lo cual puede

ser una ventaja, pero también resulta muy restrictivo cuando la instanciación depende de pointcuts complejos y se requiere mayor control. En los casos de pointcuts complejos se requiere además de una gran coordinación manual de estas situaciones. Otra desventaja de AspectJ es que es estático y requiere del código fuente. JasCo logra una mejor implementación de las conexiones entre RN y funcionalidad base, y supera a AspectJ en los puntos indicados La principal limitación de JasCo es que no incorpora la facilidad de introducciones y que no es una herramienta que ha logrado una penetración en la industria del desarrollo de software. A pesar de estas conclusiones, los autores dejan el análisis abierto a otros posibles soportes de implementación.

- A partir de los elementos de las conexiones y sus relaciones, se identifican patrones aspectuales genéricos y sus variaciones en la implementación de conexión de RN [8]. Este aporte, resulta sustantivo dado que se comprueban las dependencias que se producen en su implementación.

1.4 Conclusiones

El encapsulamiento de las conexiones entre RN y funcionalidad base con herramientas POA programáticas (AspectJ y JasCo) han demostrado sus posibilidades y limitaciones. En este sentido, una hipótesis ha trabajar es estudiar un enfoque POA declarativo en cuanto a su poder para implementar conexiones de manera flexible.

Una conexión entre una RN y la funcionalidad base es el código que invoca al objeto que representa a la RN, el código relacionado con la obtención de la información que la RN requiere, y el código que resuelva la interacción ante la posible simultaneidad de aplicación de RN. El código que implementa las conexiones entre RN y funcionalidad base resulta complejo y costoso de manejar. La reutilización y mantenimiento del software se tornan difíciles, si se considera que en un sistema se tendrá un conjunto dínámico de RN, el cual requiere actualizaciones que se suceden con mayor frecuencia que el resto del código. Manejar esta complejidad en forma manual desde el código es un problema para el desarrollador. Una estrategia más adecuada para manejar esta complejidad es abstraerla en especificaciones de alto nivel y analizarla a través de una taxonomía que ayude a la implementación.

2 OBJETIVOS DEL PROYECTO

Una conexión entre una RN y la funcionalidad base es el código que invoca al objeto que representa a la RN, el código relacionado con la obtención de la información que la RN requiere, y el código que resuelva la interacción ante la posible simultaneidad de aplicación de RN. Es a partir de estas características, que se debe plantear un enfoque para encapsular la conexión de RN con aspectos. A la vez, se debe considerar que por naturaleza, la dinámica y estructura de las RN es cambiante, lo cual exige necesariamente diseñar mecanismos de soporte flexibles. Entendiendo por flexibilidad:

a) *facilidad de reutilización*: la conexión aspectual puede ser aplicada para conectar el mismo evento de la funcionalidad base con distintas RN; o la conexión aspectual puede ser aplicada para conectar la misma RN con varios eventos;

b) *facilidad de mantenimiento*: modificaciones en la RN no afectan la conexión aspectual.

Este proyecto retoma contribuciones de otros autores con el objetivo general de explorar, proponer y experimentar estrategias y mecanismos para la conexión de RN con la funcionalidad base, flexibles que superen los resultados alcanzados.

Las bases del trabajo se establecen en los siguientes objetivos específicos:

a) plantear el abordaje de enfoques AOP declarativos para el soporte de implementación de conexiones;

b) definir una especificación de alto nivel para la representación de las conexiones independientes;

c) establecer una taxonomía que clasifique a las conexiones de acuerdo a sus características a partir de las especificaciones.

3. METODOLOGÍA

Los objetivos planteados trazan inicialmente tres ejes de abordaje:

Implementación Declarativa de Conexiones

Dado que se han estudiado en profundidad AspectJ y JasCo, se propone un enfoque que provea esencialmente soporte POA declarativo; que tenga penetración en la industria de desarrollo de software; que soporte el manejo de las conexiones de manera dinámica. Por estos motivos, se presenta en primera instancia Spring AOP Framework [25] como potencial candidato. La conjugación de soporte AOP basado en XML para definición de conexiones en forma independiente de las RN y el poder del patrón Inyección de Dependencia ofrecen diversas posibilidades. Sin embargo el mecanismo de tejido basado en Proxys plantea seguramente otras limitaciones. Estas deben ser estudiadas y analizadas para finalmente concluir el nivel de cumplimiento. También se estudiará el soporte previsto por otras herramientas de reciente aparición como Guice y GluonJ.

Especificación de Conexiones de Alto Nivel

El desarrollo de una especificación de alto nivel, abstracta e independiente de lenguajes y notaciones de diseño, permitiría capturar y representar los elementos de una RN. Esto no sólo contribuiría a su comprensión individual sino además prescinde de otros elementos que no son directamente parte del

problema. Lo cual resulta en una especificación simple y sencilla que realmente ofrezca utilidad para el desarrollador. Si además la especificación se puede construir en un soporte procesable los beneficios son aún mayores. Por eso, resulta adecuado, explorar las tecnologías XML para capturar todas las propiedades de las conexiones. A partir de [10] se identifican un conjunto básico de elementos a especificar que pueden tomarse como punto de partida para la definición de las especificaciones.

Taxonomía de Conexiones

En el plano conceptual una taxonomía que clasifique a las conexiones ayudaría a razonar sobre la complejidad de las mismas. Para ello, debe comprender y conjugar distintos niveles de análisis, relaciones de cardinalidad entre funcionalidad base y RN, tipo de información requerida por la RN, información devuelta por la RN, relaciones entre RN, etc. Esta taxonomía puede plantearse a partir de los patrones aspectuales [8]. En el plano más pragmático, la taxonomía debe poder ser aplicada a la especificación XML en forma automática. También resulta sumamente deseable que la clasificación de conexiones incorpore los métodos de implementación particulares a las herramientas POA. Esto implica que se ha creado un mecanismo de mapeo asociado.

Por lo expuesto, la metodología de investigación es esencialmente experimental, consistiendo de los siguientes pasos básicos:

- Relevamiento bibliográfico y análisis de las contribuciones relacionadas a la problemática.

- Desarrollo y ejecución de experimentos que permitan analizar el soporte de implementación de conexiones entre reglas de negocio y funcionalidad base con herramientas POA declarativas. Comparación de los resultados con otros enfoques POA.

- Identificación de los elementos y propiedades de las conexiones de reglas de negocio y funcionalidad base. Definición de una especificación de los mismos en XML.

- Implementación de un prototipo que permita el manejo de la especificación de conexiones en XML.

- Desarrollo de una taxonomía de conexiones.

- Extensión del prototipo para que incorpore la aplicación automática de la taxonomía de conexiones a las especificaciones de conexiones XML.

- Planteo y desarrollo de casos de estudio que demuestren experimentalmente la validez de la especificación XML, la taxonomía y el soporte de implementación (prototipo)

4. POSIBLES APORTES

Avance del conocimiento científico en el área

Este proyecto contribuirá a dos aspectos esenciales de la Ingeniería de Software: la reutilización de código y el mantenimiento del mismo. Hay que tener en cuenta que estos dos factores inciden directamente en los costos, esfuerzos y duración requeridos en el desarrollo del software. A la vez los resultados de este proyecto de investigación contribuyen al estudio del nuevo paradigma de desarrollo orientado a aspectos y sus aplicaciones. Los resultados del proyecto se publicarán en revistas y congresos científicos de la especialidad, de orden nacional e internacional.

Contribución al desarrollo de la industria del software

Las estrategias y mecanismos hallados y propuestos en este trabajo podrán ser incorporados para mejorar herramientas de desarrollo comerciales ya sea existentes como nuevas. Los resultados podrán ser transferidos tanto a empresas desarrollo de software de aplicación, como a empresas desarrolladoras de herramientas.

Formación de Recursos Humanos

En el marco de este proyecto se desarrollará una tesis de maestría y varias tesinas de grado. Además se continuará con la formación de los integrantes más jóvenes y alumnos del grado.

AGRADECIMIENTOS

Este proyecto de investigación esta acreditado y parcialmente financiado por la Universidad Nacional de la Patagonia Austral.

REFERENCIAS

[1] Arsanjani, A. (2001). Rule object 2001: A pattern language for adaptive and scalable business rule construction. Technical report, IBM T.J. Watson Research Centre.

[2] Binkley D., Ceccato M., Harman M., Ricca F., Tonella P. (2005), “Automated Refactoring of Object Oriented Code into Aspects”, In Proceedings of the 21st IEEE Int. Conf. on Software Maintenance (ICSM). IEEE Computer Society. USA.

[3] BRG (2001). Defining Business Rules: What Are They Really? Business Rule Group, http://www.businessrulesgroup.org/.

[4] Ceccato M., Tonella P. (2004), “Measuring the Effects of Software Aspectization”. In Cd-rom Proceedings of the 1st Workshop on Aspect Reverse Engineering (WARE 2004). The Netherlands.

[5] Chitchyan R., Rashid A., Sawyer P., Bakker J., Garcia A., Pinto Alarcon M., Tekinerdogan B., Clarke S., Jackson A. (2005), "Survey of Aspect-Oriented Analysis and Design", AOSD-Europe Project Deliverable No: AOSD-Europe-ULANC-9. Editor(s): R. Chitchyan, A. Rashid.

[6] Cibrán M. (2007) "Connecting High-Level Business Rules with Object-Oriented Applications: An approach using Aspect-Oriented Programming and Model-Driven Engineering" Tesis doctoral. Universiteit Brussel.

[7] Cibrán, M. and D'Hondt, M. (2003). "Composable and reusable business rules using AspectJ". In Workshop on Software engineering Properties of Languages for Aspect Technologies (SPLAT) at the International Conference on AOSD. Boston, USA.

[8] Cibrán M. and D'Hondt M. "A Slice of MDE with AOP: Transforming High-Level Business Rules to Aspects". (2006) In International Conference on Model Driven Engineering Languages and Systems (MODELS'06), Italy. LNCS Springer.

[9] Cibrán M. and D'Hondt M. "High-level specification of business rules and their crosscutting connections." (2006) In 8th International Workshop on AOM at the 5th International Conference on Aspect-Oriented Programming (AOSD'06), Germany,

[10] Cibrian, M., D'Hondt, M., & Jonckers, V. (2003). "Aspect-oriented programming for connecting business rules". In Proceedings of the 6th International Conference on Business Information Systems. Colorado, USA.

[11] Cibrán, M., D'Hondt, M., Suvee, D., Vanderperren, W. and Jonckers, V.(2005) "Linking Business Rules to Object-Oriented software using JAsCo#. Journal of Computational Methods in Sciences and Engineering, pp 13-27, IOS Press, Volume 5(1).

[12] Dijkstra, E.W., (1976) "A Discipline of Programming", Prentice-Hall.

[13] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). "Design Patterns, Elements of Reusable Object-Oriented Software". Addison-Wesley.

[14] Hannemann, J. & Kiczales, G. (2002). "Design pattern implementation in Java and AspectJ." In Proceedings of the 17th Annual ACM conference on Object-Oriented Programming, Systems, Languages, and Applications(OOPSLA) (pp. 161–173).

[15] Hürsch W., Lopes C. (1995); "Separation of Concerns". Northeastern University Technical Report NU-CCS-95-03, Boston.

[16] Kellens A., De Schutter K., D´Hondt T., Jonckers V. and Doggen H. (2008) "Experiences in modularizing business rules into aspects" ICSM 24 th. IEEE International Conference on Software Maintenance 2008 Page(s):448 – 451. China.

[17] Kiczales G., Lamping L., Mendhekar A., Maeda C., Lopes C., Loingtier J., Irwin J., "Aspect-Oriented Programming" (1997). In Proceedings ECOOP'97 – Object-Oriented Programming, 11th European Conference, Finland, Springer-Verlang.

[18] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W. (2001). "An overview of AspectJ". In Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP '01).

[19] Laddad R. (2003). "AspectJ in Action" Manning Publications Co.

[20] Nordberg III, M. E. (2001). "Aspect-oriented dependency inversion." In Workshop on Advanced Separation of Concerns, OOPSLA.

[21] Piveta E., Zancanela L. (2003), "Aspect Weaving Strategies", Journal of Universal Computer Science, vol.9, no. 8.

[22] Sampaio A., Loughran N., Rashid A., Rayson P. (2005), "Mining Aspects in Requirements", Workshop on Early Aspects, AOSD.

[23] Tarr P., Ossher H., Harrison W. and Sutton Jr. S.M. (1999) "N Degrees of Separation: Multi-Dimensional Separation of Concerns". Proceedings of ICSE'99. pp 107-119, IEEE Computer Society Press / ACM Press.

[24] Vanderperren, W., Suvee, D., Cibrán, M., Verheecke, B. and Jonckers, V. "Adaptive Programming in JAsCo". (2005) In Proceedings of AOSD, ACM Press, Chicago, USA

[25] Walls C. and Breidenbach R. (2004) "Spring in Action" Manning Publications Co.