Conflictos entre Aspectos en Etapas del Desarrollo de Software

Sandra Casas[1], Verónica Vanoli[1], Claudia Marcos[2] y Eugenia Marquez[1]

[1]Universidad Nacional de la Patagonia Austral. Unidad Académica Río Gallegos
Lisandro de la Torre 1070. CP 9400. Río Gallegos, Santa Cruz, Argentina
Tel/Fax: +54-2966-442313/17.
E-mail: {vvanoli, lis}@uarg.unpa.edu.ar

[2]ISISTAN Research Institute. Facultad de Ciencias Exactas. UNICEN
Paraje Arroyo Seco. CP 7000. Tandil. Buenos Aires. Argentina
Tel/Fax: + 54–2293–440362/3.
E-mail: cmarcos@exa.unicen.edu.ar

Resumen

La ausencia de métodos, estrategias y mecanismos automáticos y flexibles para la detección y resolución de conflictos entre aspectos puede tener consecuencias graves en la ejecución del software. Ante la presencia de determinadas situaciones conflictivas el comportamiento del software se puede tornar impredecible, indeseado e incierto. La construcción y desarrollo de soluciones que superen estas deficiencias resulta una tarea importante para el desarrollo orientado a aspectos [1]. El presente artículo describe el tópico de estudio, los trabajos relacionados al tema y las acciones a realizar a través del proyecto de investigación Conflictos entre Aspectos.

1. Trabajos Previos

En el año 2005 se formó un grupo de investigación formado por docentes-investigadores de la UNPA y del ISISTAN, en la línea de investigación Programación Orientada a Aspectos, específicamente sobre el tópico "Conflictos entre Aspectos". El proyecto de investigación "Estrategias para la Resolución de Conflictos en AspectJ" (29/A173) propuso el estudio de los diferentes tipos de conflictos entre aspectos y el planteo de métodos y estrategias de resolución de los mismos en particular para el lenguaje orientado de aspectos AspectJ [2]. El objetivo definido entonces fue el diseño y construcción de una herramienta que administre los conflictos. El resultado del trabajo fue la herramienta ASTOR. ASTOR [3] incorpora una serie de dispositivos que mejoran el tratamiento de conflictos en AspectJ. Los mismos están soportados por un componente Administrador de Conflictos que cumple principalmente con las funciones de detectar automáticamente conflictos y aplicar estrategias de resolución más amplias que las que AspectJ tiene por defecto, en forma semiautomática. La detección de conflictos actúa por una clasificación de los mismos por niveles de semejanza y la resolución se efectúa siguiendo las directrices de una taxonomía que proporciona seis categorías de resolución. La implementación del prototipo esta basada en el pre-procesamiento de código AspectJ, siendo además éste el único requisito para su uso. La continuidad de la línea de investigación esta dirigida a ampliar el marco de estudio mediante el proyecto "Conflictos entre Aspectos". Este proyecto tiene por objetivo hallar soluciones más flexibles y automáticas para el tratamiento de conflictos en la etapa de captura de requerimientos e implementación del desarrollo del software.

2. Introducción

La Programación Orientada a Aspectos (POA) [4] es un nuevo paradigma para el desarrollo de software que proporciona mecanismos y abstracciones para la implementación de los requerimientos transversal (Seguridad, Logging, Autentificación, Persistencia, Concurrencia, etc.) de manera separada y aislada. Es decir, la orientación a aspectos, es una técnica que permite aplicar el principio de "Separación

de Concern" [5] [6] y de esta forma, obtener una mayor y mejor modularización del código. El enfoque resulta muy prometedor y atrayente ya que supera las características indeseables producida por el efecto de "tiranía de la descomposición dominante" [7] 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 el desarrollo de software resulta más fácil de diseñar, codificar, mantener y reusar.

Un nuevo tópico es considerado como área de investigación y problemática del paradigma a resolver [1], "*el fenómeno de los conflictos entre aspectos"*, también denominados en la literatura como "*interacciones*" [8] o "*interferencias*" [9]. Un conflicto puede ocurrir cuando dos o más aspectos compiten por su activación [10]. En términos de codificación, se reconoce que un objeto de funcionalidad básica puede ser asociado a más de un aspecto, cada uno de los cuales tiene su propio objetivo de comportamiento. Si los aspectos son ortogonales [11], el sistema se ejecutará sin problemas. Pero, el comportamiento del sistema se torna impredecible si los aspectos que compiten no son independientes. En estos casos, el desarrollador debe ser informado y poder controlar estas potenciales situaciones para determinar la ejecución deseada de acuerdo al tipo de conflicto y/o el dominio de la aplicación, determinando las prioridades y políticas de activación de los aspectos.

El proyecto "Conflictos entre Aspectos" tiene por objetivo, precisamente diseñar y desarrollar métodos, mecanismos y estrategias automáticas y flexibles que den soluciones tanto a la detección como a la resolución de conflictos en diferentes etapas del desarrollo de software.

3. Herramientas POA y Manejo de Conflictos

En la mayoría de las herramientas POA, la identificación y resolución de conflictos es una tarea absolutamente manual [12]. Concretamente, si dos o más aspectos presentan una potencial situación de conflicto, el tejedor de aspectos procede sin ningún tipo de inconveniente, ni aviso y/o comunicación previa al desarrollador. Tal es el caso de AspectJ, AspectC++ [13], PHPAspect [14], etc.

En la Tabla 1 se indican brevemente los trabajos y aportes relacionados específicamente con la problemática de detección y resolución de conflictos entre aspectos.

Propuesta	Detección	Resolución
[8] [15]	Análisis de conflictos estáticos	Soporte lingüístico para la resolución de conflictos.
[16]	Detección y análisis de las interferencias causadas en AspectJ.	
[17]		Mejora el modelo de precedencia de AspectJ
[9]	Provee análisis de las interferencias aspecto-aspecto para AspectJ	
[18]	Analiza las interacciones entre aspectos. Utiliza la técnica Programme Slicing	
[19]	Sigue el esquema propuesto por [15]. Se limita a una aproximación estática de la interacción de aspectos, solo se detectan las interacciones que no ocurren en tiempo de ejecución.	Dos formas de resolver una interacción (1) elegir de las interacciones el aspecto que se va a aplicar en la ejecución (2) ordenar y anidar los aspectos para la ejecución.
[20]	Análisis de interacciones entre aspectos. Utiliza Filtros de Composición.	
[21]	Exploración inicial basada en lógica donde los hechos y reglas son definidos para la detección de interacciones en Reflex.	
[22]		Enfoque declarativo basado en restricciones para especificar la composición de aspectos

		ante un conflicto. Las restricciones pueden ser de orden o control, y pueden aplicarse en forma independiente y no combinarse.

Tabla 1: Características de los trabajos relacionados a la detección y resolución de conflictos.

4. Tratamiento de Conflictos en la Ingeniería de Requerimientos

La Ingeniería de Requerimientos Orientada a Aspectos (AORE) [23] intenta proveer soporte para la identificación y separación de los propiedades funcionales y no funcionales. En esta fase del desarrollo los requerimientos transversales se denominan aspectos tempranos (early aspects) [24]. Otro objetivo de la AORE es proveer mejores medios para la identificación y manejo de conflictos que surjan entre los aspectos tempranos, lo que se ha denominado como conflictos tempranos (early conflicts) [25].

En la Tabla 2 se sintetizan los trabajos que se refieren a los aspectos tempranos y el tratamiento de las situaciones conflictivas que se generan a partir de ellos.

Propuesta	Detección	Resolución
Requerimientos Orientados a Aspectos con UML [26]	Manual	Prioridad
Ingeniería de Requerimientos Orientada a Aspectos [27]	Manual	Prioridad/Inhabilitación/Sincronización
Especificación y Separación de Concerns desde los Requerimientos al Diseño [28]	Framework/Manual	Prioridad/Refinamiento
PROBE [29]	Framework/Manual	Orden de Preferencia y Debilidades/Cambio de Requerimientos Aclaración: Se resuelve antes de la etapa de implementación
Detección de Conflictos entre Crosscutting Concerns, basado en el Modelado [30]	Herramienta/Manual	No propone clasificación alguna Aclaración: Se resuelve en la etapa de implementación
Aspects Extractor [31]	Automática	No propone clasificación alguna

Tabla 2: Características de los trabajos relacionados a la detección y resolución de conflictos tempranos.

Existen dos puntos comunes en los enfoques citados: (i) aunque algunos de estos enfoques presentan herramientas para identificar aspectos tempranamente, la tarea que concierne al tratamiento de conflictos es totalmente manual. Esto hace que el desarrollador se haga cargo absolutamente de la toma de decisiones, no existiendo así un mecanismo automático, ni semiautomático, que ayude a resolver los conflictos sin la intervención de los desarrolladores. (ii) la inexistencia de una amplia clasificación para la resolución de conflictos. No se ofrecen alternativas de resolución. En su mayoría, se proponen niveles de prioridades para que los conflictos sean resueltos en un orden específico. Y para destacar, muchos de estos enfoques establecen información interesante en la descripción de los requerimientos, no aprovechada como material para resolver conflictos.

5. Conclusiones y Trabajo Futuro

En los apartados anteriores se ha dimensionado la relevancia de la problemática de conflictos entre aspectos en las etapas de desarrollo de captura de requerimientos e implementación. Se observa en primer lugar, que la mayoría de los enfoques y herramientas ofrecen mecanismos muy restringidos para la resolución de conflictos. Concretamente, la mayoría de éstas ofrece como resolución un enfoque que consiste en un mecanismo de orden, prioridad o precedencia. En segundo lugar, detectar la presencia y existencia de conflictos es una tarea prácticamente manual,

por estar ausente en la mayoría de las herramientas de forma automática.

La construcción y desarrollo de métodos, estrategias y mecanismos automáticos y flexibles que superen estas deficiencias y falencias resulta una tarea importante para el desarrollo orientado a aspectos [1]. En este sentido el trabajo actual se encuentra centrado en el desarrollo de las siguientes soluciones:

- Diseño y construcción de una herramienta POA que utiliza reglas explícitas y simbólicas para la detección y resolución de conflictos.
- Diseño y construcción de una herramienta para la administración temprana de conflictos.
- Detección y resolución de conflictos en código AspectJ compilado.
- Detección y resolución de conflictos en AspectJ utilizando agentes de software.

6. Formación de Recursos humanos

Tres integrantes del proyecto han enmarcado sus tesis de posgrado en la temática. El estado actual de la mismas es: (i) una tesis doctoral en etapa de escritura (ii) una tesis de maestría en estado de demostración y (iii) una tesis de maestría en estado de formulación. Los integrantes dirigen varias tesinas de grado referidas al tema y cuatro alumnos de las carreras de Licenciatura en Sistemas de la UNPA participan en el proyecto.

En el marco del proyecto los integrantes han dictado cursos de extensión, posgrado y de créditos tanta en la UNPA como en el ISISTAN.

El presente trabajo fue parcialmente financiado por la Universidad Nacional de la Patagonia Austral, Santa Cruz, Argentina.

7. Referencias

[1] Homepage of AOSD: http://aosd.net/

[2] Homepage de AspectJ Xerox, PARC, USA http://aspectj.org/.

[3] Casas S., Marcos C., Vanoli V., Reinaga H., Saldivia C., Pryor J. y Sierpe L. "ASTOR: Un Prototipo para la Administración de Conflictos en AspectJ", XIII ECC, JCC 2005, Chile.

[4] Kiczales G., Lamping J., Mendhekar A., Maeda C., Lopes C., Loingtier J., Irwin J. "Aspect-Oriented Programming". In Proceedings of ECOOP. 1997.

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

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

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

[8] Duoence R., Fradet P., Südholt M. "Detection and Resolution of Aspect Interactions", Technical Report Nº4435, INRIA, ISSN 0249-6399, Francia, 2002.

[9] ROOTS. "LogicAJ – A Uniformly Generic and Interference-Aware Aspect Language". 2005. http://roots.iai.uni-bonn.de/researh/logicaj/.

[10] Pryor J., Diaz Pace A., Campo M. "Reflection on Separation of Concerns". RITA. Vol.9. Num.1. 2002.

[11] Kienzle J., Yu Y., Xiong J. "On Composition and Reuse of Aspect", Foundations of Aspects Oriented Languages, FOAL 2003, USA.

[12] Casas S., Vanoli V., Reinaga H., Sierpe L., Pryor J., Saldivia C. "Clasificación y Resolución de Conflictos entre Aspectos" VII WICC. Río Cuarto, Argentina. 2005. ISBN: 950-665-337-2

[13] Homepage de AspectC++: http://www.aspectc.org/.

[14] Homepage of phpAspect: http://phpaspect.org/wiki/doku.php

[15] Duoence R., Fradet P., Südholt M. "A Framework for the Detection and Resolution of Aspect Interaction", In Proceeding of GPCE 2002, vol. 2487 of LNCS, USA, 2002, Springer Verlag, pp 173-188.

[16] Storzer M., Krinkle J. "Interference Analisys for AspectJ", FOAL: Foundations of Aspect-Oriented Languages, USA, 2003.

[17] Yu Y., Kienzle J. “Towards an Efficient Aspect Precedence Model”, Proceeding of the DAW, pp 156-167, England, 2004.
[18] Monga M., Beltagui F., Blair L. “Investigating Feature Interactions by Exploiting Aspect Oriented Programming”, Technical Report N comp-002-2003, Lancaster University, Inglaterra, 2003. http://www.com.lancs.ac.uk/computing/aop/Publications.php
[19] Tanter E., Noye J. “A Versatile Kernel for Multi-Language AOP”, Proceeding of ACM SIGPLAN/SIGSOFT – Conference on GPCE. LNCS, Springer-Verlag, Estonia, 2005.
[20] Durr P., Staijen T., Bergmans L., Aksit M. “Reasoning about semantic conflicts between aspects”. In K. Gybels, M. D’Hondt, I. Nagy, and R. Douence, editors, 2nd European Interactive Workshop on Aspects in Software, 2005.
[21] Kessler B, Tanter E. “Analyzing Interactions of Structural Aspects”. Workshop AID in 20th. ECOOP. France, 2006.
[22] Nagy I., Bergmans L., Aksit M. “Composing Aspects at Shared Join Point”. Workshop AID in 20th. ECOOP. France, 2006.
[23] Sampaio A., Loughran N., Rashid A., Rayson P. “Mining Aspects in Requirements”, Workshop on Early Aspects, AOSD 2005.
[24] Homepage Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design http://www.early-aspects.net/
[25] Vanoli V., Marcos C. “Administración Temprana de Conflictos entre Aspectos”. III WISBD. XII CACIC. Universidad Nacional de San Luis. Potrero de los Funes, San Luis. Octubre 2006. ISBN 950-609-050-5.
[26] Araújo J., Moreira A., Brito I., Rashid A. “Aspect-Oriented Requirements with UML”. Workshop: Aspect-oriented Modelling with UML. Dresden, Germany. October 2002.
[27] Brito I. “Aspect-Oriented Requirements Engineering”. UML. Lisbon, Portugal. October 2004.
[28] Kassab M., Constantinides C., Ormandjieva O. “Specifying and Separating Concerns from Requirements to Design: A Case Study”. International Multi-Conference on Automation, Control, and Information Technology, Anaheim, California, USA: ACTA Press. pp. 18-27. 2005
[29] Katz S., Rashid A. “From Aspectual Requirements to Proof Obligations for Aspect-Oriented Systems”. International Conference on RE, Japon, IEEE Computer Society Press. Pp 48-57, 2004.
[30] Tessier F., Badri M., Badri L. “A Model-Based Detection of Conflicts Between Crosscutting Concern: Towards a Formal Approach”. International WAOSD. China, 2004.
[31] Haak B., Díaz M., Marcos C., Pryor J. “Aspects Extractor: Identificación de Aspectos en la Ingeniería de Requerimientos”. IDEAS’06 9° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software. La Plata, Argentina, 2006.