Detección y Estimación de Conflictos en AspectJ

Sandra I. Casas[1], Marcela Maurell[1], Claudia Marcos[2]

[1]Unidad Académica Río Gallegos, Universidad Nacional de la Patagonia Austral,
Río Gallegos, Argentina, 9400

[2]Instituto de Sistemas de Tandil, Universidad Nacional del Centro
Tandil, Argentina, 7000

lis@uarg.unpa.edu.ar, cmarcos@exa.unicen.edu.ar

Resumen. *En el desarrollo de software orientado a aspectos, dos o más aspectos pueden plantear un conflicto. En ciertos casos esta interacción provoca un comportamiento impredecible y contradictorio del software. Este trabajo propone un método de detección y estimación de riesgo de conflictos negativos específica para AspectJ. El modelo esta basado en la estructura de los pointcuts y las primitivas que permiten capturar el contexto de ejecución de los join-points. La simulación del modelo permite experimentar la validez del enfoque.*

Abstract. *In Aspect-Oriented Software Development two or more aspects can outline a conflict. In certain cases this interaction causes an ambiguous behavior of the software. This work proposes a specific method to detect the conflicts and estimate the risks of the conflicts for AspectJ. The model is based on the structure of the pointcuts and the primitives which allow to capture the execution context of the join-points. The simulation of the model validates this approach.*

1. Introducción

La Programación Orientada a Aspectos [Kiczales et all., 1997] (POA) es un nuevo paradigma para el desarrollo de software que proporciona mecanismos y abstracciones para la implementación de los crosscutting concerns de manera separa y aislada. Dos o más aspectos pueden cortar transversalmente el mismo módulo de funcionalidad básica. Esta situación se conoce como "*el fenómeno de los conflictos entre aspectos"*, también denominados como "*interacciones*" [Tanter and Noye, 2005] o "*interferencias*" [Katz and Rashid, 2004]. En ciertos casos estas interacciones, pueden provocar un comportamiento impredecible del software. De esta manera, surgen los conflictos entre aspectos y la necesidad de construir mecanismos y estrategias para su adecuado tratamiento. El presente trabajo aborda la problemática de conflictos entre aspectos y propone un método de detección y estimación de riesgos de conflictos negativos específica para AspectJ [Kiczales et all., 2001].

2. Conflictos entre Aspectos en AspectJ.

Un conflicto puede ocurrir cuando dos o más aspectos compiten por su activación [Pryor et all., 2002]. Al analizar las potenciales situaciones conflictivas que pueden ocurrir en particular en AspectJ, se identifica dos tipos de conflictos, que resulta conveniente realizar su distinción. Esta clasificación responde a la estructura y semántica de los pointcuts. Los conflictos de semejanza total son evidentes y explícitos a diferencia de los conflictos de semejanza parcial, que denotan una relación entre los advices más oscura. Así, los conflictos identificados corresponden al nivel de semejanza total o al nivel de semejanza parcial, los que a continuación se explican. Se establece que existe un potencial conflicto de semejanza total sí dos o más aspectos definen pointcuts, cuyos cortes coinciden en tipo de corte y join-points, los cuales además están asociados al mismo tipo de advice (Figura 1).

```
aspect A {
  pointcut p1(): call(void C.m());
  before(): p1()
  {     ........     }
}
```

```
aspect B {
  pointcut p2(): call(void C.m());
  before(): p2()
  {      ........     }
}
```

Figura 1. Conflicto de Semejanza Total entre los aspectos A y B.

Se establece que existe un potencial conflicto de semejanza parcial sí dos o mas aspectos definen pointcuts, cuyos join-points coinciden, los cuales además están asociados al mismo tipo de advice (Figura 2).

```
aspect A  {
  pointcut p1(): call(void C.m());
  before(): p1()
    {     ........       }
 }
```

```
aspect B  {
  pointcut p2(): execution(void C.m());
  before(): p2()
    {     ........      }
}
```

Figura 2. Conflicto de Semejanza Parcial entre los aspectos A y B.

3. Estimación de Riesgos de Conflictos

El proceso de detección de conflictos tiene por objetivo la identificación de conflictos y la generación de información útil para su análisis y resolución (pointcuts y join-points, aspectos, etc.). Pero los datos utilizados en este proceso, también pueden ser empleados para estimar el riesgo del conflicto, y así aportar información más valiosa al conflicto detectado. Un modelo de estimación de riesgos de conflictos se puede establecer a partir de la aceptación de ciertos supuestos que se dan como válidos como punto de partida.

3.1 Supuestos

Las operaciones de detección y resolución de conflictos adquieren mayor importancia cuando el conflicto produce un comportamiento anómalo del software. En estos casos el conflicto se clasifica como negativo. Las posibilidades de que un conflicto sea negativo aumentan cuando los pointcuts exponen el contexto de los join-points, para que estos sean manipulados en los advices. De esta forma, si los aspectos modifican el estado del contexto (objetos, argumentos) de manera no coordinada y/o desordenada durante la ejecución del software, ciertas variables pueden asumir un estado impredecible e inestable, durante la ejecución.

3.2 Modelo de Estimación

La declaración de un pointcut provee suficiente información para establecer un modelo de estimación de riesgo y un proceso asociado de cálculo automático. En AspectJ existen 3 formas de exponer el contexto. Para cada una de estas formas existe una primitiva específica (*target, this y arg*). Entonces de acuerdo a la estructura del pointcuts se puede determinar el valor del factor de influencia del contexto. Este indicador se denomina *fic*. En la Tabla 1, se definen los valores que puede asumir el *fic* de cualquier pointcut.

Primitivas de exposición del contexto utilizadas por el pointcut p	*fic(p)*	Uso de contexto
(…)	0	Nulo
(..target (..)..)	1	Parcial
(..this (..)…)	1	Parcial
(…args (..)..)	1	Parcial
(..target (..) && arg (..)..)	2	Total
(..this (..) && arg (..)..)	2	Total

Tabla 1. Posibles valores de fic.

Esto es: sí el pointcut en análisis no utiliza las primitivas para capturar el contexto, el valor de *fic* es 0. Si se utiliza sólo una de las primitivas, el valor de *fic* es 1. Si se utilizan dos primitivas el valor de *fic* es 2. De esta forma, $0 \leq fic\ 0 \leq 2$.

Un conflicto *C* esta compuesto por un conjuntos de pointcuts, $p_1, p_2, ..p_n, (n \geq 2)$. Por cada uno de los p_i, se calcula el fic_i. El riesgo de conflicto negativo, *rcn,* es el valor promedio de los *fic*, es decir, $rcn = (fic_1 + fic_2, + .. + fic_n) / N$.

De acuerdo a los supuestos establecidos, cuanto mayores sean los valores de los *fic*, mayor será el valor de *rcn*. El valor de *rcn* esta condicionado a la captura del contexto por parte de los pointcuts. Cuando $rcn = 0$, es claro que ningún pointcuts utiliza el contexto. De manera similar ocurre cuando $rcn = 2$, significa que todos los pointcuts usan el contexto de forma total, siendo este caso el más negativo. El problema se plantea cuando $0 > rcn > 2$, significa que se utiliza solo una parte del contexto posible. Este problema no es menor, ya que la mayoría de los conflictos tendrán un *rcn* que corresponda a este rango. En la Tabla 2, se establece para los posibles valores de *rcn* el nivel de riesgo de conflicto negativo. Esta clasificación, se hace atendiendo al criterio y supuestos previamente definidos.

rcn	**Riesgo Conflicto Negativo**
0	NULO
$0 < rcn \leq 0,5$	BAJO
$0,5 < rcn \leq 1$	MEDIO-BAJO
$1 < rcn \leq 1,5$	MEDIO
$1,5 < rcn < 2$	MEDIO-ALTO
2	ALTO

Tabla 2. Niveles de Riesgo de Conflicto Negativo.

3.3 Proceso automático de cálculo.

La declaración de los pointcuts se estructura en 3 partes:

```
        (a)                (b)                          (c)
pointcut p(..):   call(void A.m(..))   &&   target(A)   &&   args(..)
```

Donde: (a) identificación del pointcut, (b) captura del join-point y (c) captura del contexto del join-point. La información de (b) se utiliza para detectar el conflicto y la información de (c) para calcular el *fic* del pointcut. Esta información puede ser obtenida mediante pre-procesamiento del código de los aspectos. Los pointcuts de los aspectos son mapeados en una tabla que contiene la siguiente información:

```
PointcutTable = {(ASPECT, POINTCUT, PRIMITIVE-CUT, JOIN-POINT, ADVICE, FIC)}
```

Por ejemplo, el mapeo del aspecto "Asp" en la tabla dinámica se efectúa básicamente como se indica en la Figura 3. En este caso el mapeo crea 3 entradas en la tabla dinámica, uno por cada pointcut (p1, p2 y p3). Aquí los pointcuts se componen de sólo de un corte primitivo (call, execution, set) sobre join-points en forma completa (void A.metodoX(), void A.metodoY(), int A.atributoX). Por cada pointcut, se ha calculado el *fic* y se ha registrado en la tabla.

```
aspect Asp {
   pointcut p1 (int i): call (void A.methodX(int) && args (i));
   pointcut p2 (A obj, int i): execution void A.methodY(int) && this(obj) &&
   arg(i));
   pointcut p3 : set int A.atributeX());
   before(int i) : p1(i)
   {   …  }
   void around(A obj, int i) : p2(i)
   {  …  }
   after() : p3()
   {  …. }
}
```

Asp	p1	call	A	methodX	before	1
Asp	p2	execution	A	methodY	around	2
Asp	p3	set	A	atributeX	after	0
…						
…						

Figura 3. Mapeo del aspecto Asp en la tabla de pointcuts.

En AspectJ un pointcut se puede definir por la composición de varios cortes primitivos (utilizando los constuctores "&&", "||" y "!"), además los join-points se pueden definir por comprensión, mediante el uso de comodines. En estos casos, el mapeo de un aspecto no será tan lineal como el ejemplo, es decir un join-point puede producir más de una entrada en la tabla de pointcuts. Previamente al mapeo, cada pointcut es analizado y se determina el conjunto de entradas que produce en la tabla.

Luego que todos los pointcuts han sido mapeados en la tabla, un algoritmo verifica la existencia de conflictos de semejanza total o semejanza parcial. En el caso de detectar la ocurrencia de un conflicto se calcula el *rcn* del mismo. El cómputo de *rcn* es inmediato y automático.

4. Simulaciones, pointcuts y conflictos

El análisis del método de estimación propuesto es complejo si se pretende aplicar a un conjunto considerable de aplicaciones AspectJ reales. Por esta razón, se realiza una experiencia simulatoria. En cada escenario de la simulación se consideró una cantidad de join-points (100) y aspectos (5) delimitada, la distribución de los pointcuts fue homogénea. Sobre cada uno de ellos se trazó un número estipulado pero variable de pointcuts (30, 60, 90 y 120). La cantidad de pointcuts (np) puede resultar un tanto

exagerada, pero a los efectos de la simulación, estos pointcuts son simples (no se utilizan comodines de ningún tipo). Por cada escenario, se generaron 50 muestras (cada muestra representa una aplicación POA en AspectJ). Cada muestra es única. Los datos que se generaron en cada muestra, corresponden a los pointcuts de la misma. La construcción de la declaración de cada pointcuts es totalmente automática y aleatoria (uso de cortes primitivos call o execution, uso de las primitivas de contextos, asignación de un advice, asignación de un aspecto y join-point). Por cada pointcuts generado se procede a su mapeo en la tabla de pointcuts y se calcula el *fic* asociado. Luego que se han generado los *np* pointcuts de la muestra, se continúa con la detección de conflictos. Cada vez que se identifica un conflicto, se determina el tipo y se calcula *rcn* (los conflictos pueden estar formados por cualquier cantidad de pointcuts). Se continúa de esta forma, hasta que todos los pointcuts de la muestra hayan sido analizados. Los valores obtenidos se registran convenientemente, antes de comenzar con la próxima muestra. Se considera que cada escenario es comparable con los otros, ya que los procesos de formación o generación de pointcuts que pudieron afectar las distribuciones son semejantes.

Join-points = 100	Aspectos = 5				M = 50	
np = 30	Conflictos detectados			Frecuencias		
Rcn	C	CST	CSP	*fic=0*	*fic=1*	*fic=2*
rcn=0	3	3	0	100	0	0
0 < rcn ≤ 0,5	12	6	6	50	50	0
0,5 < rcn ≤ 1	27	15	12	11.11	77.78	11.11
1 < rcn ≤ 1,5	18	8	10	0	50	50
1,5 < rcn < 2	0	0	0	0	0	0
rcn=2	3	2	1	0	0	100
np = 60	Conflictos detectados			Frecuencias		
rcn	C	CST	CSP	*fic=0*	*fic=1*	*fic=2*
rcn=0	9	3	6	100	0	0
0 < rcn ≤ 0,5	65	24	41	50.38	49.62	0
0,5 < rcn ≤ 1	108	53	55	13.39	73.66	12.95
1 < rcn ≤ 1,5	67	35	32	0	50.37	49.63
1,5 < rcn < 2	2	2	0	0	33.33	66.67
rcn=2	19	15	4	0	0	100
np = 90	Conflictos detectados			Frecuencias		
rcn	C	CST	CSP	*fic=0*	*fic=1*	*fic=2*
rcn=0	37	17	20	100	0	0
0 < rcn ≤ 0,5	140	77	63	50.18	49.82	0
0,5 < rcn ≤ 1	207	91	116	17.35	67.35	15.3
1 < rcn ≤ 1,5	146	70	76	0.99	49.67	49.34
1,5 < rcn < 2	4	1	3	0	33.33	66.67
rcn=2	30	11	29	0	0	100
np = 120	Conflictos detectados			Frecuencias		
rcn	C	CST	CSP	*fic=0*	*fic=1*	*fic=2*
rcn=0	59	22	37	100	0	0
0 < rcn ≤ 0,5	219	124	95	50.56	49.44	0
0,5 < rcn ≤ 1	377	195	182	16.54	68.69	14.77
1 < rcn ≤ 1,5	248	119	129	0.59	50.2	49.22
1,5 < rcn < 2	6	3	3	0	33.33	66.67
rcn=2	68	37	31	0	0	100

Tabla 3. Resultados de la simulación.

La Tabla 3 resume los resultados obtenidos por la simulación. Los valores correspondientes a los conflictos (C) se expresan en unidades y estos se discriminan en conflictos de semejanza total (CST) y conflictos de semejanza parcial (CSP). Los valores correspondientes a las frecuencias de los *fic* están expresados en forma porcentual. Al

mantener constate la cantidad de aspectos y join-point y en forma sucesiva aumentar la cantidad de pointcuts, se fuerza la ocurrencia de situaciones conflictivas en forma creciente. De este modo, se puede analizar si los resultados obtenidos son casuales o responden a una propiedad o característica regular. En este caso, se observa en todos los escenarios (independientemente del valor de np): la mayoría de los conflictos tiene $0 <$ rcn <2, entre el 87% y 91% de los conflictos tiene un riesgo MEDIO (uso parcial del contexto); cuando $0 <$ rcn $\leq 0,5$, los pointcuts que hacen uso del contexto tienen fic=0 y fic=1 en similar proporción (en todos los casos entre el 49% y 51%); cuando $0,5 <$ rcn ≤ 1, alrededor del 70% de los pointcuts en conflicto tienen fic = 1, los pointcuts con fic = 0 y fic = 2 son insignificantes; cuando $1 <$ rcn $\leq 1,5$, los pointcuts que hacen uso del contexto tiene fic = 1 y fic = 2 en similar proporción (en todos los casos entre el 49% y 51%); cuando $1,5 <$ rcn < 2, la mayoría de los pointcuts en conflicto usan el contexto forma total, ya que les corresponde un fic = 2. La experiencia permitió conocer con precisión las frecuencias de los *fic* a partir de distribuciones de pointcuts y los conflictos detectados. Esta circunstancia es la que posibilita evaluar si los criterios para determinar los niveles de conflicto negativo son válidos.

5. Conclusiones

El presente trabajo propone un método de detección y estimación de conflictos entre aspectos, especifico para AspectJ. La declaración de los pointcuts es la única información que se requiere para el proceso, posibilitando que éste sea totalmente automático y simultáneo. El proceso de identificación reconoce dos tipos de conflictos y la estimación de riesgo de conflictos negativos se basa en la información del contexto de la ejecución de los join-points. Los riesgos de conflictos negativos, son clasificados de acuerdo al uso del contexto, pudiendo ser nulo, bajo, medio-bajo, medio, medio-alto y alto. Esta información ayuda al desarrollador en la posterior resolución de conflictos. Un experimento simulatorio ha demostrado que el modelo de estimación es válido y razonable.

El presente trabajo fue parcialmente financiado por la Universidad Nacional de la Patagonia Austral, Santa Cruz, Argentina y el proyecto PICT 32079 (ANPCYT).

Referencias

Kiczales G., Lamping L., Mendhekar A., Maeda C., Lopes C., Loingtier J., Irwin J. (1997), “Aspect-Oriented Programming”. In Proceedings ECOOP’97.

Tanter E., Noye J. (2005), “A Versatile Kernel for Multi-Language AOP”, Proceeding of ACM SIGPLAN/SIGSOFT – Conference on Generative Programming and Component Engineering LNCS, Springer-Verlag, Estonia.

Katz S., Rashid A. (2004), “From Aspectual Requirements to Proof Obligations for Aspect-Oriented Systems”, International Conference on Requirements Engineering, Japon, IEEE Computer Society Press. Pp 48-57.

Kiczales G., Hilsdale E., Hugunin J., Kersten M., Palm J., Griswold W. (2001), “An Overview of AspectJ”. ECOOP 2.001.

Pryor J., Diaz Pace A., Campo M. (2002), “Reflection on Separation of Concerns”. RITA. Vol.9. Num.1.