Administración Temprana de Conflictos entre Aspectos

Verónica Laura Vanoli
Unidad Académica Río Gallegos, Universidad Nacional de la Patagonia Austral
Río Gallegos, Santa Cruz, Argentina
e-mail: vvanoli@uarg.unpa.edu.ar

and

Claudia Marcos
ISISTAN Research Institute, Facultad de Ciencias Exactas. UNICEN
Tandil, Buenos Aires, Argentina
e-mail: cmarcos@exa.unicen.edu.ar

Abstract

Due to constant changes on the aspect oriented software development, it has been necessary to apply this paradigm in all the stages of the software development. That explains the current works which include the earliest stages, existing approaches that identify aspects from functional and non-functional crosscutting concerns. Some Works deal with treatment of conflicts between these aspects. Given that these works are in an east treatment, this work proposes an automated approach to administer conflicts in the early stages of the software development, precisely, in the requirements engineering; in order to detect, analyze and solve conflicts that may arise among aspects generated in early stages.

Keywords: Early-stages, Early-aspects, Candidate aspects, Early-conflicts.

Resumen

Dada la creciente evolución y cambios existentes en el desarrollo de software orientado a aspectos se ha visto necesario aplicar dicho paradigma en todas las etapas del desarrollo de software. Es por ello que actualmente existen trabajos que abarcan hasta las etapas más tempranas, existiendo enfoques que identifican aspectos a partir de los crosscutting concerns funcionales y no funcionales. Inclusive, en algunos casos abordan el tratamiento de conflictos entre dichos aspectos. Debido a que recién se inicia este tratamiento, este trabajo propone un enfoque automatizado para administrar conflictos en las etapas tempranas del desarrollo de software, más precisamente en la ingeniería de requerimientos, de manera tal de detectar, analizar y resolver conflictos que puedan surgir entre aspectos generados tempranamente.

Palabras clave: Etapas tempranas, Aspectos tempranos, Aspectos candidatos, Conflictos tempranos.

1 INTRODUCCION

El Desarrollo de Software Orientado a Aspectos (DSOA) [6] apunta al manejo de crosscutting concerns proveyendo mecanismos para su identificación, separación, representación y composición sistemática. Los crosscutting concerns se encapsulan en módulos separados conocidos como aspectos, para promover la localización de los mismos. Con ello resulta ser un mejor soporte para la modularización reduciendo así, los costos de desarrollo, mantenimiento y evolución de sistemas.

En el DSOA es frecuente encontrar situaciones en las que un componente de funcionalidad básica se encuentra afectado por dos o más aspectos. Cada uno de los cuales le adiciona diferente comportamiento al componente funcional. Esta situación puede provocar un comportamiento impredecible o indeseado en el sistema, generando así, lo que se conoce como conflicto. Es decir, un conflicto puede ocurrir cuando dos o más aspectos compiten por su activación [11].

Hasta el momento existen propuestas que se enfocan de diferentes formas para tratar los conflictos, particularmente en las últimas etapas del desarrollo de software. Entre ellos se puede nombrar a: AOP/ST [2], Alpheus [12] y Astor [4].

Dado que los sistemas evolucionan y se encuentran en constante cambio, es de suma importancia la modularización durante el proceso completo de desarrollo de software, llegando incluso hasta la etapa de ingeniería de requerimientos. En ella, no sólo se identifican y especifican los crosscutting concerns (llamados aspectos candidatos) sino que también se puede detectar y resolver las posibles competencias de activación que surjan a partir de dos o más aspectos candidatos.

Debido a que la tendencia en el DSOA hacia las etapas tempranas es relativamente nueva y aún más el trato que se le da a los conflictos, es por ello que este trabajo propone un enfoque para administrar automáticamente conflictos en la ingeniería de requerimientos. La administración consistirá de tres tareas esenciales como detectar, analizar y resolver posibles situaciones conflictivas que surjan a partir de los aspectos candidatos generados en las etapas tempranas del desarrollo de software.

El trabajo se estructura de la siguiente manera. En la Sección 2 se introduce y explican las etapas tempranas en el desarrollo de software. En la Sección 3 se describen los trabajos relacionados a este enfoque. En la Sección 4 se describe el trabajo en su totalidad, desde el proceso utilizado, niveles de conflictos, y clasificación y resolución aplicados; y finalmente, en la Sección 5 se exponen las conclusiones.

2 ETAPAS TEMPRANAS EN EL DESARROLLO DE SOFTWARE

El hecho de identificar los aspectos en etapas tempranas del ciclo de desarrollo de software orientado a aspectos significa la reducción de costos de desarrollo, mantenimiento y evolución de los sistemas [7]. Inclusive si se refiere a la ingeniería de requerimientos, ya que se reduce el peligro de cambios no esperados en etapas posteriores en los productos de software. Si acompañado a todo esto también se resuelven tempranamente los conflictos, el resultado es, sin ninguna duda, aún mucho mejor. Esto se debe a que las decisiones tomadas tempranamente, respecto a las situaciones conflictivas, no repercuten de la misma manera que si son tomadas posteriormente.

A continuación, se desprenden dos conceptos fundamentales del desarrollo de sistemas en etapas tempranas analizados por separado y en detalle para dar comienzo a este trabajo. Ellos son: los aspectos tempranos y los conflictos tempranos.

2.1 Aspectos Tempranos

En el nuevo paradigma del DSOA, los distintos crosscutting concerns se identifican y separan desde el principio. Esta división se mantiene y evoluciona con las propias especificaciones y módulos durante todo el proceso de desarrollo. Por lo tanto, se ha dado en llamar aspectos tempranos (early-aspects) a los que son identificados durante las etapas iniciales del proceso de desarrollo, en especial la captura de requisitos y el diseño arquitectónico [5].

Con la aplicación de aspectos tempranos se logra establecer criterios básicos y estandarizados con respecto a los aspectos. Esto quiere decir, que desde los comienzos del desarrollo de software hasta su implementación, los aspectos no sufrirán grandes alteraciones respecto a su funcionalidad. Si bien, no se puede evitar que en etapas posteriores los aspectos tengan un tratamiento diferente que a los comienzos, se evitarán problemáticas que surjan desde que el cliente solicita un sistema, mejorando la adaptación del mismo a los cambios de requerimientos.

En la actualidad existen varios trabajos relacionados a tratar la identificación y especificación de crosscutting concerns en etapas tempranas del ciclo de desarrollo de software, como se describe en la Sección 3. Un enfoque particular [7] está conformado por una serie de tareas automatizadas, donde se analiza la información provista por el usuario y se extraen datos de interés que llevarán a la obtención de los aspectos candidatos. Finalmente, se necesita la decisión del analista, quien seleccionará los aspectos definitivos.

2.2 Conflictos Tempranos

Puede ocurrir que dos o más aspectos compitan por su activación, es decir, intenten cortar transversalmente, al mismo tiempo, una determinada funcionalidad básica del sistema. De esta manera se genera una interferencia entre dichos aspectos. Una interferencia puede ser conflictiva o no. Puede suceder que el corte no afecte al normal desempeño del sistema, y así no provocar ninguna anomalía en la ejecución de los aspectos. Este es el caso de una interferencia no conflictiva denominada contribución positiva. Caso contrario ocurre si dos o más aspectos alteran el comportamiento del sistema al intentar acceder al mismo tiempo. En este caso, se denomina contribución negativa y es necesario administrarla. Este trabajo detecta ambas contribuciones.

Desde el punto de vista de la ingeniería de requerimientos orientada a aspectos, se ha dado en llamar conflictos tempranos (early-conflicts) a toda contribución negativa que se genera a partir de la competencia de activación entre aspectos tempranos.

Administrar tempranamente los conflictos entre aspectos reduce considerablemente el trabajo que implica identificarlos y resolverlos exclusivamente en la etapa de implementación, es decir, en etapas posteriores o tardías del ciclo de desarrollo de software. Además, se preveen con anterioridad comportamientos ajenos al normal desarrollo del sistema. Y así, lograr la toma de decisiones con los stakeholders lo antes posible y no esperar a que el sistema se encuentre casi finalizado. Es muy importante aclarar que de esta manera se logra que la administración de conflictos también sea una tarea a desarrollar en todas las etapas del ciclo de vida del software.

3 TRABAJOS RELACIONADOS

Varios trabajos se han desarrollado para el estudio de aspectos en las etapas tempranas del desarrollo de software, que hacen referencia a situaciones conflictivas que generan los aspectos identificados tempranamente. A continuación se describen estos trabajos.

Araújo y otros [1] proponen resolver los conflictos, durante la etapa de ingeniería de requerimientos, a través de prioridades entre aspectos candidatos. Se descuidan situaciones como qué ocurre si existen aspectos candidatos con igual prioridad y si al resolverlos, surgen nuevos conflictos.

Brito [3] completa la clasificación de resolver por prioridades (proponiendo niveles de prioridades), con la inhabilitación de ciertos aspectos candidatos y una sincronización de los mismos (algo así como concurrencia). La sincronización no deja en claro su resolución y para los niveles de prioridades no se propone un límite de nivelación, abusando así de ellos. Se involucra a los stakeholders para la toma de decisiones, de quienes se necesita mucha precisión para negociar los cambios. Se resuelven los conflictos en la etapa de ingeniería de requerimientos de software.

Kassab y otros [9] mejoran la parte de aspectos tempranos a través de un framework, pero la clasificación sólo abarca resolución por prioridades. Otra alternativa de resolución consta de un proceso de refinamiento de requerimientos, pero queda poco claro cómo es y quién participa en él. Se provee muy buena información para los requerimientos funcionales, pero la misma, no es aprovechada para el tratamiento de conflictos. Se resuelven los conflictos en la etapa de ingeniería de requerimientos de software.

PROBE [10] es un framework donde se establece una interesante subdivisión de conflictos, como nivelación previa a la resolución. Pero dicha subdivisión resulta ser dependiente del framework. Se propone como clasificación un orden de preferencia entre aspectos candidatos, estableciendo debilidades. También se presenta una solución por medio del cambio en los requerimientos, que no resulta clara y poco recomendable. Si fuere así, estos cambios pueden ocasionar otros posibles conflictos, en cuyo caso tampoco se encuentra contemplado. Involucran también a los stakeholders para la toma de decisiones. Los conflictos se resuelven antes de la etapa de implementación.

Tessier y otros [13] proveen una categorización detallada de los conflictos y dos niveles de conflictos: directos (aspectos candidatos con conexiones directas) e indirectos (aspectos candidatos con conexiones más difíciles de detectar). Sólo tratan el primero de los niveles. No existe clasificación para la resolución de conflictos y se deja la posible solución para la etapa de implementación.

La herramienta Aspects Extractor [8] es un completo enfoque automatizado para identificar, especificar, integrar y evaluar los crosscutting concerns en la etapa de ingeniería de requerimientos de software. Proporcionando un conjunto de tareas que se automatizan con el objetivo de asistir al analista en la identificación de aspectos candidatos. Solamente se identifican los conflictos. Una sugerencia como propuesta de clasificación para la resolución de los mismos es por medio de la priorización de los aspectos candidatos. Se deja la resolución para ser tratada posteriormente por el analista, no siendo parte de la herramienta.

Existen dos puntos principales a tener en cuenta que son comunes a los enfoques descriptos anteriormente.

En primer lugar, 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 stakeholder se haga cargo absolutamente de la toma de decisiones, no existiendo así un mecanismo automático, ni mucho menos semiautomático, que ayude a resolver los conflictos sin la intervención de los stakeholders.

En último lugar, es la no existencia 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.

4 ENFOQUE AUTOMATIZADO PARA ADMINISTRAR CONFLICTOS

En este trabajo se propone automatizar por completo la administración de conflictos. Para ello se toma como base la herramienta *Aspects Extractor* (descripta en la Sección anterior) y se le adiciona la posibilidad de resolver cualquier situación conflictiva entre aspectos candidatos que la herramienta identifica.

La herramienta *Aspects Extractor* define, en su modelo de ingeniería de requerimientos, la Tarea 4 titulada *Identificar conflictos*. La función principal esta tarea es la de identificar los conflictos que se generan cuando dos o más aspectos candidatos se aplican sobre un mismo caso de uso. En este trabajo se aplica una extensión a dicha tarea para adaptarla a la administración de conflictos. La Figura 1 describe el conjunto de tareas propuestas para identificar o detectar, analizar y resolver las posibles situaciones conflictivas.

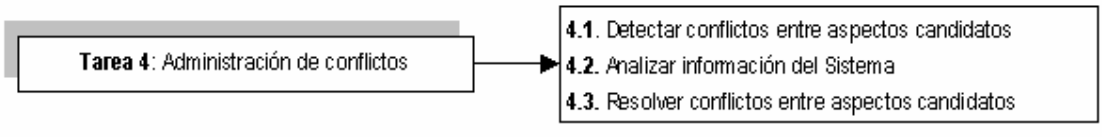

Figura 1: Extensión del Modelo de Ingeniería de Requerimientos de *Aspects Extractor*.

La Tarea 4 se denomina ahora Administración de conflictos y la misma consta de las siguientes tres Subtareas:

. Subtarea 4.1, se encarga de la detección (identificación) de conflictos.
. Subtarea 4.2, se analiza la información del Sistema necesaria para resolver los conflictos.
. Subtarea 4.3, se resuelven los conflictos detectados.

El mecanismo por completo se realiza de forma sistemática, y cada subtarea depende de la anterior para cumplir con su funcionalidad.

4.1 Proceso de Administración de Conflictos

La Figura 2 muestra en detalle el proceso propuesto para administrar tempranamente los conflictos en este trabajo. El proceso contiene componentes cuya definición constituye el eje principal de la propuesta automatizada de este enfoque.

- *Conflictos entre Aspectos Candidatos*: es la entrada de datos que contiene los aspectos candidatos que generan situaciones conflictivas.
- *Buscar en Repositorio Conflictos Evidentes*: es la tarea que se encarga de determinar la existencia o no de conflictos evidentes ya recopilados en un repositorio. Es decir, aquellas situaciones conflictivas resueltas en aplicaciones anteriores.
- *Repositorio Conflictos Evidentes*: es la base de información donde se almacenan las resoluciones de conflictos exitosas.
- *Analizar Casos de Uso*: es todo dato que surja de la descripción de los casos de uso. Servirá como información en este componente para tomar decisiones de resolución a todo conflicto que se genere.
- *Interactuar con Stakeholders*: cuando no exista posibilidad alguna para resolver conflictos de forma automatizada, los stakeholders podrán aportar información necesaria para la resolución. Con el tiempo cada vez será menos necesaria su intervención.
- *Resolver Conflictos*: es la tarea más importante de este trabajo, es donde se ofrece una solución a las situaciones conflictivas.

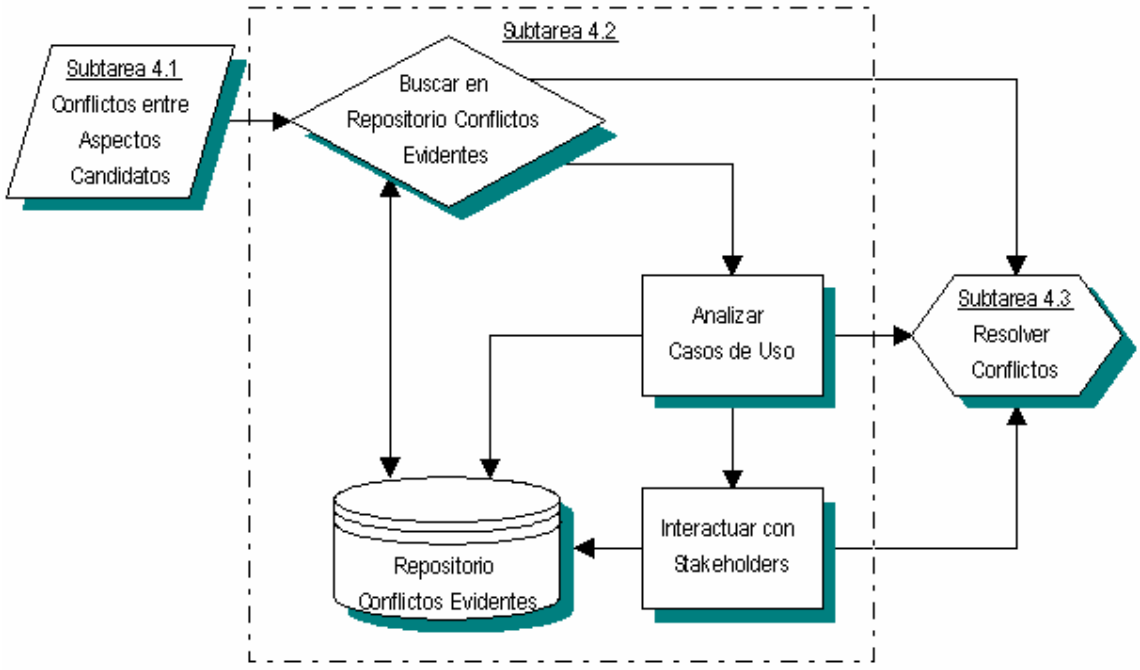

Figura 2: Proceso de Administración Temprana de Conflictos entre Aspectos

El flujo del proceso de administración temprana de conflictos tiene como entrada una lista de conflictos generada a partir de los aspectos candidatos identificados por la herramienta *Aspects Extractor* (Subtarea 4.1).

La zona marcada con líneas punteadas representa a la Subtarea 4.2, donde se analiza la información para la toma de decisiones respecto a cómo se van a tratar los conflictos, para su posterior resolución. En dicho análisis, se realiza en primer término la búsqueda de conflictos evidentes en el repositorio. Si la búsqueda resulta exitosa, automáticamente la situación conflictiva pasa a ser resuelta. Caso contrario, es decir, la búsqueda fracasa, se procede a analizar la descripción y propiedades de los casos de uso, por los cuales los aspectos candidatos en conflicto compiten por su activación. Respecto a los casos de uso, ya sea la descripción, las precondiciones y poscondiciones, las suposiciones, los flujos básico y alternativo, etc. pueden ser una fuente útil para efectuar el análisis. Como resultado del mismo puede ocurrir que el caso de uso no proporcione o no contenga información necesaria como para una correcta toma de decisión, en este caso, se hace partícipe al stakeholder que corresponda, permitiendo una interactividad donde la toma de decisión queda a cargo del mismo. Caso contrario, si el análisis resulta favorable, es decir, se puede rescatar información del caso de uso, primero se actualiza el repositorio de acuerdo a la información rescatada y luego se procede a la resolución automática del conflicto.

La Subtarea 4.3, es la resolución de los conflictos identificados, marcando a los aspectos candidatos en conflicto, de forma tal que los mismos queden registrados para proseguir con el desarrollo del sistema. Esto se verá reflejado en el diagrama de casos de uso extendido modelado con UML, donde se identificarán los conflictos a través de un lenguaje formal y la representación visual estará conformada por nuevas relaciones entre los aspectos candidatos conflictivos. En la Subsección 4.3 se presenta la clasificación adoptada para resolver dichos conflictos.

4.2 Niveles de Conflictos

Es conveniente realizar una distinción de los conflictos a través de niveles, basada en las dos clases de contribuciones vistas en la Subsección 2.2 (positiva y negativa), que se puedan encontrar en las distintas situaciones conflictivas.

A continuación se describen cuatro niveles de conflictos propuestos:

- **Interferencia libre**: es toda aquella competencia de activación entre aspectos candidatos que no genera ningún tipo de conflicto. Es decir, es la ausencia de conflictos. Si bien los aspectos candidatos compiten por su activación, no se afectan unos a los otros. Este sería el caso de una contribución positiva.
- **Conflicto evidente**: es toda aquella situación conflictiva conocida, es decir, ya ha sido detectada con anterioridad y presenta su posible solución. Para este caso, se cuenta con un repositorio que contiene los aspectos candidatos con sus respectivas situaciones conflictivas ya conocidas. Dicho repositorio, principalmente se actualizará automáticamente en el transcurso del tiempo con su utilización, pero también puede existir una intervención manual ya sea con anterioridad o durante su uso. Los campos principales del repositorio son los aspectos candidatos, el conflicto generado e identificado entre ellos, el nivel de conflicto al que pertenece y la resolución aplicada en tal caso. El contenido de los campos pasa a ser una sugerencia por definición.
- **Conflicto avanzado**: es toda aquella competencia de activación entre nuevos aspectos candidatos identificados que no se encuentra en el repositorio de conflictos evidentes. Por lo tanto, luego de la administración que corresponda a tal situación, se actualiza el repositorio cargando los nuevos aspectos candidatos y la situación conflictiva generada por los mismos.

- **Conflicto intermedio**: es toda aquella competencia de activación entre nuevos aspectos candidatos identificados que no se encuentra en el repositorio de conflictos evidentes y aspectos candidatos que ya se encuentren almacenados en el mismo. Por lo tanto y luego de la administración correspondiente, se actualiza el repositorio cargando el nuevo aspecto candidato y la nueva conexión con el aspecto candidato existente en el repositorio.

4.3 Clasificación y Resolución de Conflictos

La clasificación que se propone para la resolución de conflictos en la etapa temprana del desarrollo de software, se encuentra ampliada y adaptada de acuerdo a la taxonomía de resolución [12]. Hasta el momento se ha identificado una clasificación de diez tipos de políticas de resolución en caso de activación de conflictos: orden, orden inverso, orden natural, orden automático, exclusión, nulidad, dependiente del contexto, partición, unión y combinaciones.

A continuación se presentan los diez tipos nombrados:

- Orden. Se establece un orden para los aspectos candidatos en conflicto, planteando prioridades para su activación.
- Orden inverso. Se establece un orden invertido para los aspectos candidatos en conflicto. De la misma manera que el caso anterior, sólo que las prioridades planteadas se invierten.
- Orden natural. Se establece un orden según como la herramienta *Aspects Extractor* devuelve listado a los aspectos candidatos en conflicto.
- Orden automático. En los casos donde ninguna de las políticas anteriores pueda ser aplicada, se determina un orden por medio de un mecanismo de aleatoriedad.
- Exclusión. Se determina que sólo un aspecto candidato en conflicto quedará incluido en el modelo, por lo tanto, el resto de los aspectos candidatos serán excluidos.
- Nulidad. Se determina la nulidad total de los aspectos candidatos en conflicto. De esta manera, todos los aspectos candidatos quedarán excluidos del modelo.
- Dependiente del contexto. Toma partición en este caso todo comportamiento externo a los casos de uso, para influir en la decisión de resolución.
- Partición: Se determina que un aspecto candidato en conflicto pueda ser divido al menos en dos partes, generando así la posibilidad de crear nuevos aspectos candidatos que no generen situaciones conflictivas.
- Unión: Se determina que dos o más aspectos candidatos en conflicto puedan unificarse al menos en un solo aspecto candidato y así resolver la o las situaciones conflictivas.
- Combinaciones. Cuando una política de resolución no sea suficiente como para resolver el conflicto, se puede combinar con otras políticas y así lograr el objetivo deseado.

5 CONCLUSIONES

En este trabajo se presenta un enfoque automatizado extendido de la herramienta *Aspects Extractor*, a la que se le incorpora una nueva tarea. La misma consiste en la administración de conflictos que se generan a partir de crosscutting concerns en las etapas tempranas del desarrollo de software, más precisamente en la Ingeniería de Requerimientos. La administración consiste en detectar, analizar y resolver posibles situaciones conflictivas.

Es muy importante administrar conflictos entre aspectos tan pronto como sea posible puesto que permite la aplicación orientada a aspectos desde el comienzo del ciclo de vida del software. Permitiendo así también, la extensión a todo el proceso de desarrollo. De esta manera, se evitan futuras correcciones difíciles de detectar. Y se reduce considerablemente los costos de desarrollo, mantenimiento y evolución.

Se continúa trabajando en la propuesta. A medida que se avance en la utilización del enfoque automatizado, éste ira enriqueciendo el repositorio de conflictos evidentes y así, se reducirá considerablemente la intervención del stakeholder para interactuar en la toma de decisiones para resolver las situaciones conflictivas.

6 AGRADECIMIENTOS

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

REFERENCIAS

[1] Araújo J., Moreira A., Brito I., Rashid A. "Aspect-Oriented Requirements with UML". Workshop: Aspect-oriented Modelling with UML. Dresden, Germany. October 2002

[2] Boellert K. "On Weaving Aspects". Proceeding of the Aspect-Oriented Programming Workshop at ECOOP'99. Lisbon, Portugal. June 1999

[3] Brito I. "Aspect-Oriented Requirements Engineering". UML 2004. Lisbon, Portugal. October 2004

[4] Casas S., Marcos C., Vanoli V., Reinaga H., Saldivia C., Pryor J., Sierpe L. "ASTOR: Un Prototipo para la Administración de Conflictos en AspectJ". XIII Encuentro Chileno de Computación (ECC 2005). Jornadas Chilenas de Computación - UACh (JCC'05). Valdivia, X Región, Chile. 7 al 12 de Noviembre de 2005

[5] Cuesta C., Romay P., de la Fuente P., Solórzano M. "Aspectos como Conectores en Arquitectura de Software". II Jornadas DYNAMICA. España. Noviembre de 2004

[6] Elrad T., Filman R., Bader A. "Theme Section on Aspect-Oriented Programming". Communications of ACM, Vol. 44 No. 10. 2001.

[7] Haak B., Díaz M., Marcos C., Pryor J. "Identificación Temprana de Aspectos". V Workshop de Ingeniería de Software (WIS'05). Jornadas Chilenas de Computación - UACh (JCC'05). Valdivia, X Región, Chile. 7 al 12 de Noviembre de 2005

[8] 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. Facultad de Informática, Universidad Nacional de La Plata. 24 a 28 de Abril de 2006

[9] 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

[10]Katz A., Rashid A. “PROBE: From Requirements and Design to Proof Obligations for Aspect-Oriented Systems”. Computing Department Lancaster University Technical Report Number: COMP-002-2004. February 2004

[11]Pryor J., Diaz Pace A., Campo M. “Reflection on Separation of Concerns”. RITA. Vol 10, Num 2. 2002

[12]Pryor J., Marcos C. “Solving Conflicts in Aspect-Oriented Applications”. Proceeding of the Fourth ASSE. 32 JAIIO. Universidad Argentina de la Empresa. 1 al 5 de Septiembre de 2003

[13]Tessier F., Badri M., Badri L. “A Model-Based Detection of Conflicts Between Crosscutting Concerns: Towards a Formal Approach”. International Workshop on Aspect-Oriented Software Development (WAOSD). Peking University, Beijing, China. September 2004