Una Taxonomía para Conflictos Tempranos

Verónica Vanoli[1] y Claudia Marcos[2]

[1] Unidad Académica Río Gallegos. Universidad Nacional de la Patagonia Austral
Lisandro de la Torre 1070. CP 9400. Río Gallegos. Santa Cruz. Argentina
Tel/Fax: +54-2966-442313/17. vvanoli@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. cmarcos@exa.unicen.edu.ar

Resumen. En estos últimos años, el tratamiento de situaciones conflictivas entre aspectos aplicado en la etapa de implementación en el desarrollo de un sistema, se ha trasladado a etapas más tempranas, más precisamente a la Ingeniería de Requerimientos. La resolución de conflictos en dicha etapa aún carece de un tratamiento extensivo y cuidadoso. Es por ello que este enfoque propone una amplia taxonomía de resolución como clasificación de tipos de conflictos detectados, para ser resueltos de forma clara, ordenada y diversificada. Aportando de esta manera un enriquecimiento al tratamiento de conflictos en etapas tempranas del desarrollo de software.

Palabras claves: Conflictos entre aspectos tempranos, Taxonomía de conflictos entre aspectos.

1 Introducción

En los últimos años, se ha considerado el fenómeno de “conflictos entre aspectos” como nuevo tópico de investigación en el Desarrollo de Software Orientado a Aspectos (DSOA). Conocido como “interacciones” [1][2] o “interferencias” [3][4].

La ocurrencia de conflictos entre aspectos es una consecuencia de la descomposición de los sistemas de software y su posterior composición en el DSOA. Esto se debe a que una aplicación en dicho paradigma consiste esencialmente de un conjunto de unidades o componentes: clases y aspectos. Es frecuente encontrar situaciones en las que un componente de funcionalidad básica se encuentre afectado por más de un aspecto. Cada uno de ellos le adiciona diferente comportamiento provocando una interacción entre éstos. En ciertos casos esta interacción, puede tener por resultado un comportamiento contradictorio, nulo o innecesario. Así, surgen los conflictos entre aspectos y la necesidad de construir mecanismos y estrategias para su adecuado tratamiento, ya sea la identificación, análisis y resolución de los conflictos.

Dichos tratamientos deben ser independientes de las herramientas utilizadas y requieren una dedicación especial, dado que la activación de ciertos conflictos puede provocar comportamientos no deseados e inciertos en la ejecución de aplicaciones orientadas a aspectos. Los primeros tratamientos de conflictos más amplios propuestos, se han presentado a nivel de implementación, alguno de ellos se encuentran en herramientas como: Alpheus [5], AOP/ST [6], AspectC/C++ [7],

ASTOR [8], DJAspect [9], y otras. En estos últimos años, dicho tratamiento se ha trasladado a las etapas más tempranas del desarrollo de software, más precisamente a la Ingeniería de Requerimientos. Realizar un tratamiento de situaciones conflictivas entre aspectos candidatos al inicio del desarrollo reduce considerablemente el trabajo que implica identificarlos y resolverlos en etapas tardías del ciclo de vida. Además, se preveen con anterioridad comportamientos ajenos al normal desarrollo del sistema; y así, se logra que los desarrolladores tomen decisiones lo antes posible y no esperar a que se encuentre muy avanzada la construcción del sistema.

En este trabajo se propone adaptar y mejorar los mecanismos existentes para lograr una correcta administración temprana de conflictos [10]. La administración consiste 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. En la última tarea, la de resolución, se propone una amplia taxonomía, como clasificación de tipos de conflictos detectados.

Este trabajo se organiza de la siguiente manera. En la Sección 2 se profundiza en el concepto de conflictos tempranos. En la Sección 3 se describen resumidamente los trabajos relacionados. En la Sección 4 se presenta el enfoque principal de este trabajo, detallando la taxonomía según sus componentes principales, niveles de conflictos establecidos, los tipos de conflictos surgidos según análisis de información y algunos casos que ejemplifican la propuesta. Finalmente, en la Sección 5 se exponen las conclusiones y trabajos futuros.

2 Conflictos Tempranos

Definimos conflicto cuando dos o más aspectos intenten cortar transversalmente, al mismo tiempo, una determinada funcionalidad básica del sistema. Cuando dicho acceso no afecta al normal desempeño del sistema, es decir, no provoca ninguna anomalía en la ejecución de los aspectos, se dice que la situación conflictiva es una contribución positiva [11]. Este es el caso donde es necesario detectar la situación conflictiva, pero no se realiza acción alguna. Por el contrario, si dos o más aspectos alteran el comportamiento del sistema al intentar acceder al mismo tiempo una componente funcional, se denomina contribución negativa [11]. En esta contribución no sólo es necesario detectar la situación conflictiva sino que es importante resolverla.

Desde el punto de vista de la Ingeniería de Requerimientos Orientada a Aspectos (IROA) [12], definimos a los conflictos tempranos, con el título de early-conflicts, como toda contribución negativa que se genera a partir de que dos o más aspectos tempranos compitan entre sí por su activación.

Es muy importante aclarar que la detección y resolución de conflictos en etapas tempranas del desarrollo de software promueve a que dicho tratamiento sea también una tarea a desarrollar en todas las etapas del ciclo de vida.

3 Trabajos Relacionados

Existen diversos trabajos referentes al estudio de aspectos en las etapas tempranas del desarrollo de software que tratan las situaciones conflictivas generadas por los

aspectos candidatos, como lo son: Requerimientos Orientados a Aspectos con UML [13], IROA [14], Especificación y Separación de Concerns desde los Requerimientos al Diseño [15], PROBE [4], Detección de Conflictos entre Crosscutting Concerns, basado en el Modelado [16] y Aspects Extractor [17].

Luego de analizar cómo tratan estas propuestas los conflictos tempranamente, se puede determinar que: si bien algunos de estos trabajos presentan herramientas o frameworks para identificar aspectos tempranamente, la tarea que concierne al tratamiento de conflictos es básicamente manual, ignorando la posibilidad de que el desarrollador cuente con un mecanismo automático o semiautomático, para resolver las situaciones conflictivas. De esta manera, el desarrollador es responsable de las decisiones de resolución manualmente.

Por otro lado, es notable que ninguno de estos trabajos proponga o presente una amplia clasificación para la resolución de conflictos, es decir, no se ofrecen alternativas de resolución. A pesar de que generan información valiosa en la descripción de los requerimientos, no se aprovecha para el tratamiento de conflictos. Algunos de estos enfoques, proponen al desarrollador niveles de prioridades (no determinados) para que los conflictos se resuelvan en un orden especificado.

4 Taxonomía de Conflictos entre Aspectos

Para poder identificar las posibles situaciones conflictivas se considera que los mismos han sido identificados y especificados de forma similar a los casos de uso. Dados los diferentes ítems correspondientes a la especificación de la funcionalidad del sistema y por medio de sus casos de uso es posible inferir diferentes tipos de conflictos y así, su correspondiente resolución. Existen niveles de conflictos definidos que genera un mejor ordenamiento de las situaciones conflictivas detectadas.

4.1 Definición del Conjunto de Datos

El conjunto de datos iniciales con los que se cuenta para representar la taxonomía propuesta en este trabajo, consta de cuatro elementos principales (casos de uso, aspectos candidatos, conflictos y repositorio de conflictos) descriptos en lenguaje natural y en lenguaje formal. Para este último caso, en la tabla 1 se ve reflejada la representación simbólica utilizada.

Tabla 1. Tabla de Símbolos.

Símbolos	Descripción	Símbolos	Descripción
<<	Antecesor	¬	Negación
∧	Conjunción (y lógico)	∀	Para todo o Todo
{ , }	Conjunto de elementos	∈	Pertenece
[,]	Conjunto de elementos opcionales o alternativos	∴	Por lo tanto
⊇	Contiene	:=	Se define como
∨	Disyunción (o lógico)	/	Tal que
⇒	Entonces	:	Tiene o Posee o Cuenta con
∃	Existe	∪	Unión de conjunto de elementos

A continuación se detallan los cuatro elementos nombrados anteriormente según su significado y su representación.

» **Casos de Uso (CU)**

Todo caso de uso CU tiene una especificación de casos de uso que consta de los siguientes ítems: nombre, identificador, descripción, prioridad, actor, trigger, suposiciones, pre-condición, curso básico, curso alternativo, post-condición, terminación, requerimientos especiales, casos de uso que extiende, casos de uso incluidos, generalización del caso de uso y frecuencia.

El ítem prioridad posee tres puntos a examinar: nivel, importancia y urgencia. Cada uno de ellos tiene distintos grados de prioridad. Pueden existir niveles primarios, secundarios u opcionales; la importancia puede ser vital, importante o conveniente; y un caso de uso puede tener una urgencia inmediata, necesaria o puede esperar.

El ítem frecuencia establece una serie de opciones ordenadas de mayor a menor, según la ocurrencia en el sistema: continuo al ser instalado el sistema, continuo y simultaneo con el uso del sistema, continuo con el uso del sistema, aislado y regular con el uso del sistema, y aislado con el uso del sistema.

CU : Especificación(CU) $\supseteq$ ítems / ítems : {Nombre, Identificador, Descripción, Prioridad, Actor, Trigger, Suposiciones, Pre-condición, Curso Básico, Curso Alternativo, Post-condición, Terminación, Requerimientos especiales, Casos de Uso que extiende, Casos de Uso incluidos, Generalización del Caso de Uso, Frecuencia}

Items.Prioridad : {Nivel, Importancia, Urgencia}

Items.Prioridad.Nivel : [Primario, Secundario, Opcional]

Items.Prioridad.Importancia : [Vital, Importante, Conveniente]

Items.Prioridad.Urgencia : [Inmediata, Necesaria, PuedeEsperar]

Items.Frecuencia : [Continuo al ser instalado el sistema, Continuo y Simultaneo con el uso del sistema, Continuo con el uso del sistema, Aislado y Regular con el uso del sistema, Aislado con el uso del sistema]

» **Aspectos Candidatos (A)**

Todo aspecto candidato A posee una especificación de aspectos cuyos ítems son los mismos que la especificación de casos de uso más la incorporación del caso de uso que origina dicho aspecto.

A : Especificación(A) $\supseteq$ ítems / ítems : Especificación(CU) $\cup${Caso de Uso Origen}

» **Conflictos (C)**

Todo conflicto C se define de la siguiente manera, existe un aspecto candidato A_1 tal que dicho aspecto pertenece a un caso de uso CU_1 y existe otro aspecto A_2 el cual pertenece al mismo caso de uso que A_1.

$$C := \exists A_1 / A_1 \in CU_1 \wedge \exists A_2 / A_2 \in CU_1$$

Todo conflicto C tiene su identificación, una descripción, el tipo de conflicto detectado, y la resolución de conflicto aplicada.

C : {Identificador, Descripción, Tipo de Conflicto, Resolución Aplicada}

» **Repositorio de Conflictos Evidentes (R)**

El repositorio de conflictos evidentes R contiene un conjunto de conflictos evidentes acompañado de los aspectos involucrados en dicho conflicto (A_i y A_j) y la resolución aplicada a dicha situación conflictiva, exitosamente.

R : {C.Identificador, A_i, A_j, C.Resolución Aplicada}

4.2 Definición de los Niveles de Conflictos

Se han definido cuatro niveles de conflicto para los cuales se tiene en cuenta las dos clases de contribuciones descriptas anteriormente (contribución positiva y negativa):
Conflicto Evidente (*R : {C.Identificador, A_1, A_2, C.Resolución Aplicada}*): es toda situación conflictiva conocida, es decir, ha sido detectada y resuelta con anterioridad. Por lo tanto, se encuentra almacenada en el repositorio de conflictos evidentes. Este es el caso donde el análisis de la información no es necesario, dado que la resolución fue aplicada, sólo bastaría con elegirla y proseguir. A menos que el desarrollador quiera optar por otra o por intermedio de algún mecanismo automático se proporcione una mejor alternativa de resolución y así, se mantiene la automatización.

Interferencia Libre ($\vdash C$): es toda 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 y no se debe tomar ninguna medida de resolución.

Conflicto Avanzado (*R : {}*): es toda nueva competencia de activación entre aspectos candidatos que no ha sido encontrada en aplicaciones anteriores, es decir, que no son conflictos evidentes.

Conflicto Intermedio (*R : {C.Identificador, [A_1, A_2], C.Resolución Aplicada}*): es toda aquella competencia de activación entre aspectos candidatos, donde uno de ellos ha participado anteriormente en una situación conflictiva y el otro no. Es el caso donde se deberá resolver la situación conflictiva, pero teniendo en cuenta que alguno de los aspectos candidatos involucrados ya han participado en otro conflicto registrado.

4.3 Tipos de Conflictos

» **Conflicto de Orden (O):** Se ejecutan ambos aspectos en conflicto según los siguientes criterios de orden:

O1. Si uno de los aspectos A_1 se encuentra en el ítem Trigger del caso de uso CU y el otro aspecto A_2 en el ítem Terminación del caso de uso CU, se determina que es un conflicto de orden (C_1), por lo tanto, el orden de ejecución es primero el aspecto encontrado en el ítem Trigger (A_1) y segundo el aspecto encontrado en el ítem Terminación (A_2).

$$A_1 \in \text{Especificación(CU).Trigger} \wedge A_2 \in \text{Especificación(CU).Terminación} \Rightarrow \text{Orden}(C_1) \therefore \text{Resolución} : \{A_1, A_2\}$$

O2. Si uno de los aspectos A_1 se encuentra en el ítem Pre-condición del caso de uso CU y el otro aspecto A_2 en el ítem Post-condición del caso de uso CU, se determina que es un conflicto de orden (C_1), por lo tanto, el orden de ejecución es primero el aspecto encontrado en el ítem Pre-condición (A_1) y segundo el aspecto encontrado en el ítem Post-condición (A_2).

$$A_1 \in \text{Especificación(CU).Pre-condición} \wedge A_2 \in \text{Especificación(CU).Post-condición} \Rightarrow \text{Orden}(C_1) \therefore \text{Resolución} : \{A_1, A_2\}$$

O3. Si ambos aspectos (A_1 y A_2) se encuentran en el ítem Curso Básico del caso de uso CU, se determina que es un conflicto de orden (C_1), por lo tanto, el orden de ejecución es primero aquel aspecto que se encuentre ubicado antes que el otro (A_1) y en consecuencia segundo es el que se encuentre ubicado después (A_2).

$$A_1 \in Especificación(CU).Curso\ Básico \wedge A_2 \in Especificación(CU).Curso\ Básico \,/\, A_1 << A_2 \Rightarrow Orden(C_1) \therefore Resolución : \{A_1, A_2\}$$

O4. Si uno de los aspectos A_1 se encuentra en el ítem Curso Básico del caso de uso CU y el otro aspecto A_2 en el ítem Curso Alternativo del caso de uso CU, se determina que es un conflicto de orden (C_1), por lo tanto, el orden de ejecución es primero el aspecto encontrado en el ítem Curso Básico (A_1) y segundo el aspecto encontrado en el ítem Curso Alternativo (A_2).

$$A_1 \in Especificación(CU).Curso\ Básico \wedge A_2 \in Especificación(CU).Curso\ Alternativo \Rightarrow Orden(C_1) \therefore Resolución : \{A_1, A_2\}$$

» **Conflicto por Inclusión Simple (IS):** Se activa un solo aspecto de los aspectos que se encuentran en conflicto, según los siguientes criterios de inclusión:

IS1. Si uno de los aspectos A_1 se encuentra en el ítem Suposiciones del caso de uso CU y el otro aspecto A_2 en el ítem Pre-condición del caso de uso CU, se determina que es un conflicto por inclusión simple (C_1), por lo tanto, se incluye el aspecto encontrado en el ítem Pre-condición (A_1) y se excluye el aspecto encontrado en el ítem Suposiciones (A_2).

$$A_1 \in Especificación(CU).Suposiciones \wedge A_2 \in Especificación(CU).Pre\text{-}condición \Rightarrow InclusiónSimple(C_1) \therefore Resolución : \{\neg A_1, A_2\}$$

Este criterio surge de considerar que el ítem Pre-condición posee mayor importancia respecto al ítem Suposiciones, dado que las pre-condiciones son más representativas al caso de uso que las suposiciones, dado que estas últimas no son verificadas por el caso de uso. De esta manera, le correspondería al caso de uso que lo verifica considerar o no al aspecto excluido.

» **Conflicto por Nulidad (N):** Se anulan los aspectos en conflicto, no permitiendo la ejecución de ninguno de ellos, según el siguiente criterio de nulidad:

N1. Si la frecuencia de ocurrencia (ítem Frecuencia del caso de uso CU) es la más baja de todas (Aislado con el uso del sistema), se determina que es un conflicto por nulidad (C_1), por lo tanto, se anulan ambos aspectos (A_1 y A_2).

$$A_1 \wedge A_2 \in Especificación(CU).Frecuencia\ /\ Frecuencia.\ Aislado\ con\ el\ uso\ del\ sistema \Rightarrow Nulidad(C_1) \therefore Resolución : \{\neg A_1, \neg A_2\}$$

El criterio aplicado en este caso, surge de establecer que un caso de uso cuya frecuencia de ocurrencia es la mas baja de todas, es decir, absolutamente aislada (el tiempo que se espera que suceda el caso de uso no tiene una continuidad en el sistema), y en consecuencia los aspectos también, no pone en riesgo al sistema, por lo tanto, si no existe otra alternativa de resolución entre los aspectos en conflicto, anularlos podría ser una alternativa viable.

» **Conflicto en Orden Natural (ON):** Cuando no exista información de análisis como para determinar el modo de ejecución de los aspectos conflictivos (A_1 y A_2), se determina que es un conflicto de orden natural (C_1), por lo tanto, se establece un orden entre los aspectos según como hayan sido devueltos en forma listada originalmente. Es decir, permanecen en el mismo orden.

$$\{A_1, A_2\} \Rightarrow OrdenNatural(C_1) \therefore Resolución : \{A_1, A_2\}$$

Los tipos de conflictos continúan siendo estudiados y también el desarrollo de ejemplos para demostrar la factibilidad de la propuesta. Por lo tanto, continua la propuesta de nuevas formas de resolución de conflictos. Otros tipos de conflictos son:

» **Conflicto por Partición (P):** Los aspectos en conflicto sufren una división (un aspecto se transforma en dos subaspectos nuevos) generando así nuevos aspectos candidatos que deberán ser procesados nuevamente para verificar si de esta manera provocan alguna situación conflictiva nuevamente.

» **Conflicto por Unión (U):** Los aspectos en conflicto pasan a acoplarse entre sí, deviniendo en un solo aspecto.

» **Conflicto por Inclusión Compuesta (IC):** Se aplica a los casos de más de dos aspectos en conflicto. Permitiendo ejecutar más de un aspecto y excluyendo otros. Los aspectos que son incluidos, deberán ser procesados nuevamente para verificar si provocan alguna situación conflictiva.

» **Casos Especiales:** De los anteriores pueden ir surgiendo casos como por ejemplo: *O5-IS2*. Si la prioridad (ítem Prioridad del aspecto candidato A) de uno de los aspectos A_1 es superior (según una combinación del nivel, importancia y urgencia) al otro aspecto A_2, se determina que puede ser un conflicto de orden o un conflicto de inclusión simple (C_1). Por lo tanto, si el desarrollador elige el primer tipo, en el orden de ejecución es primero aquel aspecto que cumpla con dicha superioridad (A_1) y en consecuencia segundo el superado en prioridad (A_2). Caso contrario, si el desarrollador elige el segundo tipo, se incluye aquel aspecto que cumpla con dicha superioridad (A_1) y en consecuencia se excluye el superado en prioridad (A_2).

$$Especificación(A).Prioridad / A_1 : \{Nivel.Primario, Importancia.Vital, Urgencia.Inmediata\} \wedge A_2 : \{Nivel.Opcional, Importancia.Conveniente,Urgencia.PuedeEsperar\} \vee Especificación(A).Prioridad / A_1 : \{NivelSecundario, Importancia.Importante, Urgencia.Necesaria\} \wedge A_2 : \{Nivel.Opcional, Importancia.Conveniente, Urgencia.PuedeEsperar\} \vee Especificación(A).Prioridad / A_1 : \{Nivel.Primario, Importancia.Vital, Urgencia.Inmediata\} \wedge A_2 : \{Nivel.Secundario, Importancia.Importante, Urgencia.Necesaria\} \Rightarrow Orden(C_1) \therefore Resolución : \{A_1, A_2\} \vee InclusiónSimple(C_1) \therefore Resolución : \{A_1, \neg A_2\}$$

4.4 Ejemplos de Tipos de Conflictos

A continuación se pueden apreciar dos ejemplos que demuestran como llegar a los tipos de conflictos vistos anteriormente y aplicar una resolución de manera exitosa. Para cada uno, se presentan los datos principales con los que cuenta (el caso de uso y los aspectos que cortan al caso de uso, es decir, los aspectos en conflicto). Luego se explica como se llega al tipo de conflicto y por último, como se resuelve el conflicto, mostrando al conflicto con todos sus componentes. Al finalizar, se muestra el repositorio de conflictos evidentes (*R*) con las situaciones conflictivas ya cargadas. Sólo para el primer caso del primer ejemplo, se muestra parte de la especificación del caso de uso, es decir, de donde se originan los aspectos conflictivos (Tabla 2).

Ejemplo 1. Sistema Servidor de Correo electrónico
1.a) CU_1=Enviar un correo electrónico; A_1=Verificación de sesión; A_2=Seteo de datos

Tabla 2. Parte de la especificación del caso de uso CU_1.

Nombre	Enviar un correo electrónico
…	
Descripciones: **Curso Básico**	1. El caso de uso comienza cuando el usuario desea enviar un correo electrónico. 2. El sistema verifica (*crosscut* <Verificación de sesión>) que la sesión se encuentre activa. 3. El sistema muestra una pantalla donde el usuario puede setear (*crosscut* <Seteo de datos>) el destino y el contenido del mail. 4. El sistema solicita confirmación de envío. 5. El usuario selecciona la opción enviar. 6. El sistema envía el correo electrónico e informa el envío. 7. El sistema guarda en el mensaje enviado.
…	

Al ser analizada la información del caso de uso "Enviar una correo electrónico" en dicho sistema, teniendo en cuenta los aspectos candidatos en conflicto A_1 y A_2, se deduce que el aspecto A_1 y el aspecto A_2 se encuentran ubicados en el ítem Curso Básico, entonces, se toma el criterio presentado en *O3*. Es decir, es un tipo de orden. Por lo tanto, el orden de ejecución es primero el aspecto "Verificación de sesión" y segundo el aspecto "Seteo de datos".

C_1 : {C0001, "Situación conflictiva básica", "Conflicto de Orden", O3}

1.b) CU_2=Abrir un correo electrónico; A_3=Visualización de contenidos; A_4=Tiempo de respuesta

Según los aspectos candidatos en conflicto A_3 y A_4, se deduce que la prioridad del aspecto A_3 (Nivel.Primario, Importancia.Vital, Urgencia.Inmediata) es superior a la del aspecto A_4 (Nivel.Secundario, Importancia.Conveniente, Urgencia.Necesaria), entonces, se toma el criterio presentado en *O5-IS2*. Es decir, puede ser un tipo de orden o de inclusión simple. Por lo tanto, si el desarrollador opta por el segundo tipo, se incluye el aspecto "Visualización de contenidos" y se excluye el aspecto "Tiempo de respuesta".

C_2 : {C0002, "Situación conflictiva combinada", "Conflicto de Orden/Inclusión Simple", O5-IS2}

Ejemplo 2. Sistema Gestión de Alumnos
2.a) CU_1=Inscribir en una materia; A_1=Persistencia; A_2=Verificación de actividades

Al analizarse la información del caso de uso "Inscribir en una materia", teniendo en cuenta los aspectos candidatos en conflicto A_1 y A_2, se deduce que el aspecto A_2 se encuentra en el ítem Pre-condición y el aspecto A_1 se encuentra en el ítem Post-condición, entonces, el criterio tomado es *O2*. Es decir, es un tipo de orden. Por lo tanto, el orden de ejecución es primero el aspecto "Verificación de actividades" y segundo el aspecto "Persistencia".

C_3 : {C0003, "Situación conflictiva básica", "Conflicto de Orden", O2}

2.b) CU_2=Estadísticas de alumnos; A_3=Filtrado de datos; A_4=Seguridad de conexión

Según los aspectos candidatos en conflicto A_3 y A_4, se deduce que ambos aspectos requieren de su aplicación e importancia en el caso de uso, entonces, el criterio tomado es el tipo en orden natural (*ON*), dado que no es posible determinar algún criterio específico para ellos. Por lo tanto, los aspectos "Filtrado de datos" y "Seguridad de conexión" se ejecutan de acuerdo al orden en que fueron especificados anteriormente.

C_4 : {C0004, "Situación conflictiva fuerte", "Conflicto de Orden Natural", ON}
R : {{C0001, "Verificación de sesión", "Seteo de datos", O3}; {C0002, "Visualización de contenidos", "Tiempo de respuesta", O5-IS2}; {C0003, "Persistencia", "Verificación de actividades", O2}; {C0004, "Filtrado de datos", "Seguridad de conexión", ON}}

Otro caso que puede presentarse resulta en descubrir que los aspectos en conflicto ya han sido resueltos, es decir, se encuentran en el repositorio. Por lo tanto, los caminos a seguir son opcionales, o se aplica la resolución registrada o se vuelve a resolver la situación conflictiva para ver si es posible encontrar otra solución alternativa e incluso mejor. Por ejemplo, "Filtrado de datos" y "Seguridad de conexión", puede ser resuelto por *ON* (como lo indica el repositorio) o por *IS1*. Es probable que en un comienzo no resultó posible determinar algún criterio específico, entonces se aplicó un orden ya establecido, pero en una revisión posterior se descubre que este conflicto puede ser resuelto por *IS1*, descartando así la seguridad de conexión en una estadística del Sistema de Gestión de Alumnos, dado que no afecta la funcionalidad del caso de uso. En este caso, debería actualizarse el repositorio.

5 Conclusiones y Trabajos Futuros

El seguimiento de las situaciones conflictivas provocadas por los aspectos candidatos ayuda en el desarrollo, documentación y mantenimiento de las aplicaciones orientadas a aspectos. En caso de que surja algún cambio en las etapas posteriores o una vez que el sistema concluye, será posible obtener la información de los aspectos involucrados en esos conflictos y modificarlos sin necesidad de buscar en todo el sistema.

En este trabajo se presenta un enfoque para ampliar la resolución de conflictos en las etapas tempranas del desarrollo de software durante la captura de requerimientos. Dicha ampliación consiste en la presentación de una taxonomía de conflictos con una clasificación de los mismos por tipos de conflictos, según una nivelación que ordena las diferentes situaciones conflictivas. El trabajo es la base de una herramienta en desarrollo que permite administrar tempranamente situaciones conflictivas generadas por aspectos identificados en la etapa de ingeniería de requerimientos.

Se continua trabajando en la propuesta, proponiendo nuevos tipos de conflictos que ayuden a una correcta toma de decisiones a la hora de resolver una situación que provoca inestabilidad en el sistema. Adicionalmente, se están desarrollando ejemplos de manera tal de demostrar la factibilidad de la propuesta.

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

Referencias

1. Douence R., Fradet P., Südholt M. “A Framework for the Detection and Resolution of Aspect Interactions”. Proceedings of the ACM SIGPLAN SIGSOFT Conference on GPCE. (2002)
2. Tanter E., Noye J. “A Versatile Kernel for Multi-Language AOP”, Proceeding of ACM SIGPLAN/SIGSOFT – Conference on Generative Programming and Component Engineering (GPCE’05) LNCS, Springer-Verlag, Estonia, (2005)
3. Bergmans L. M. J. “Towards detection of semantic conflicts between crosscutting concerns”. In ECOOP: AAOS ’03: The first workshop on Analysis of Aspect-Oriented Software, Darmstadt, Germany, July 21. (2003)
4. 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)
5. Pryor J., Marcos C. “Solving Conflicts in Aspect-Oriented Applications”. Proceedings of the Fourth ASSE. 32 JAIIO. Buenos Aires. Argentina. ISSN 1666-1141. Septiembre (2003)
6. Boellert K. “On Weaving Aspects”. Proc. of the AOP Workshop at ECOOP. (1999)
7. Gal A., Scroder-Preikschat W., Spinczyk O. “AspectC++: Language Proposal and Prototype Implementation”. ACM International Conference Proceeding Series Proceedings of the Fortieth International Conference on Tools Pacific: Objects for internet, mobile and embedded applications. Volume 10. Sydney, Australia. (2002)
8. 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. (2005)
9. Pryor J., Bastán N., Campo M. “A Reflective Approach to Support Aspect Oriented Programming in Java”. In Proceedings of the ASSE. 29 JAIIO. Tandil. Argentina. (2000)
10. Vanoli V., Marcos C. “Early Conflicts: Análisis y Resolución de Conflictos Tempranos”. Jornadas Chilenas de Computación (JCC07). XIX Encuentro Chileno de Computación (ECC07). Iquique, Chile. 5 al 10 de Noviembre. http://prat.unap.cl/jcc2007. (2007)
11. Chung L., Nixon B., Yu E., Mylopoulos J. “Non-Functional Requirements in Software Engineering”. Kluwer Academic Publishing. (2000)
12. Sampaio A., Loughran N., Rashid A., Rayson P. “Mining Aspects in Requirements”. Workshop on Early Aspects, AOSD (2005)
13. Araújo J., Moreira A., Brito I., Rashid A. “Aspect-Oriented Requirements with UML”. Workshop: Aspect-oriented Modelling with UML. Dresden, Germany. October (2002)
14. Brito I. “Aspect-Oriented Requirements Engineering”. UML 2004. Lisbon, Portugal. October (2004)
15. 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)
16. 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)
17. 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 (2006)