Identificación y Modelado de Aspectos Tempranos dirigido por Tarjetas de Responsabilidades y Colaboraciones

Sandra Casas y Héctor Reinaga

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

Abstract

The key of the Aspect-Oriented Development is to achieve the separation of concerns along the whole process of software construction. The identification of crosscutting concerns and the modeling of the early aspects from the initial phases improve the tracing, the composition, the evolution and the scalability of the software. This work explores and it proposes a method to identify crosscutting concerns and to model early aspects by means of the use of CRC cards. The method is supported by the TAOM tool.

Keywords: crosscutting concerns, CRC cards, early aspects, AOP.

Resumen

La clave del Desarrollo de Software Orientado a Aspectos es lograr la Separación de Concerns a lo largo de todo el proceso de construcción de software. La importancia de identificar crosscutting concerns y modelar aspectos tempranos desde las fases iniciales mejora la traceabilidad, composición, evolución y escalabilidad del software. Este trabajo explora y propone un método para identificar crosscutting concerns y modelar aspectos tempranos mediante el uso de tarjetas CRC. El método esta soportado por la herramienta TAOM.

Palabras Claves: crosscutting concerns, tarjetas CRC, early aspects, AOP.

1.INTRODUCCIÓN

La clave del Desarrollo de Software Orientado a Aspectos (AOSD) [1] es alcanzar la Separación de Concerns (SoC) [2][3] a lo largo de todo el proceso de construcción de software. Un *concern* puede ser definido de manera genérica como algo de interés para un proceso de ingeniería [4]. Relacionado con este principio, normalmente se encuentra el problema de los crosscutting concerns. El término crosscutting usualmente se describe en términos de los conceptos scattering y tangling, por ejemplo, un crosscutting concern puede ser definido como un concern que se encuentra disperso (scattering) y enmarañado (tangling) con otros concerns.

Pero los síntomas scattering y tangling no sólo ocurren en los artefactos de implementación, también emergen en otros artefactos durante el proceso de desarrollo. Por esta razón, es necesario aplicar la SoC en todas las fases del ciclo de vida. Como resultado, la comprensibilidad, el mantenimiento y la reusabilidad de los artefactos de construcción de software se perfeccionan. Además la captura explícita de crosscutting concerns durante todas las fases de desarrollo hace posible a los desarrolladores tracear los crosscutting concerns desde los requerimientos a la implementación. La importancia de identificar crosscutting concerns y modelar aspectos desde las etapas tempranas de desarrollo se ha plasmado en varios trabajos [5]. En este sentido las estrategias han apuntado a la extensión y adaptación de los métodos convencionales como viewpoints, casos de uso y orientado a objetivos [6][7][8] los cuales dan soporte a la nueva abstracción, denominada "early aspect" [9].

Las tarjetas CRC [10] [11] son una técnica simple e informal pero efectiva que ha sido propuesta tanto para el modelado conceptual como para el diseño detallado de sistemas OO. Formalmente no existen antecedentes del uso de tarjetas CRC en AODS. Esencialmente este trabajo explora la aplicación y extensión de este artefacto para la identificación de crosscutting concerns y modelado conceptual de aspectos tempranos. Asimismo se plantea un proceso sencillo para razonar y comprender la naturaleza de los concerns y alcanzar la SoC. Además se presenta el prototipo "TAOM" que da soporte al método propuesto.

El presente trabajo se estructura de la siguiente manera: en la Sección 2 se rescatan brevemente los fundamentos de la Tarjetas CRC y se explora su aplicación en la identificación de crosscutting concerns y modelado de aspectos tempranos; en la Sección 3 se presenta el método propuesto; en la Sección 4 se presenta la herramienta TAOM; en la sección 5 se analiza la propuesta de acuerdo a diversos criterios; en la Sección 6 se presentan los trabajos relacionados y en la Sección 7 se exponen las conclusiones y el trabajo futuro.

2.El "porque" de las tarjetas CRC

K. Beck y W. Cunningham presentaron las tarjetas CRC (Class-Responsability-Colaborations) en 1989 para la enseñanza de la OOP [12]. Desde entonces, la técnica ha sido refinada hasta llegar a ser un valioso artefacto, no sólo para la enseñanza sino también para el modelado conceptual y/o diseño detallado de sistemas OO. El método "Diseño Dirigido por las Responsabilidades" (RDD) propuesto por Wirfs-Brock [13] [14] se basa en la definición de tarjetas CRC y contratos. Varios métodos ágiles emplean intensamente las tarjetas CRC. La "Programación eXtrema" (XP) [15] propone las tarjetas CRC como técnica de diseño y Cockburn las considera en su método "Modelado basado en Responsabilidades" (RBM) [16]. En este último caso, además se propone la complementación de las tarjetas CRC con elementos de la notación UML.

La característica más sobresaliente de las tarjetas CRC sea su simpleza y ductilidad. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema.

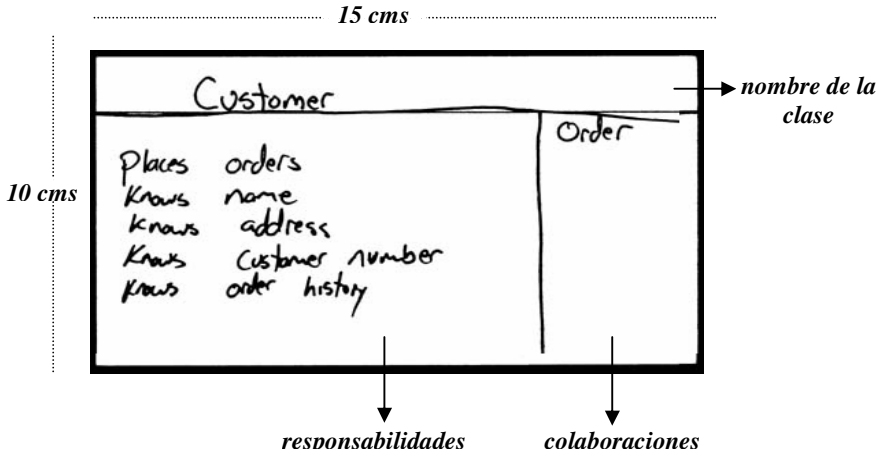

Figura 1: Tarjeta CRC Custumer.

Como se ilustra en la Figura 1, una tarjeta CRC establece 3 dimensiones las cuales identifican el rol de un objeto en análisis y/o diseño: nombre de la clase, responsabilidades y colaboraciones. Una clase representa a una colección de objetos similares. Una responsabilidad es aquello que la clase sabe o hace. Una clase puede cambiar el valor de lo que sabe pero no puede cambiar el valor de lo que saben otras clases. Algunas veces una clase tiene una responsabilidad que cumplir pero no tiene toda la información para hacerlo. Esto hace que deba interactuar con otras clases para obtener su colaboración. Las colaboraciones toman una de dos formas: un pedido de información o un pedido de que se realice una operación.

El formato físico de las tarjetas CRC facilita la interacción entre los stakeholders en sesiones en las que se aplican técnicas de grupos como "tormenta de ideas" o "juego de roles" y se ejecutan escenarios a partir de especificación de requisitos, historias de usuarios o casos de uso. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones. Luego en un estadío de diseño avanzado o ya en la implementación del sistema, las tarjetas CRC se convierten en clases con métodos y atributos.

Desde un enfoque puramente OO cada tarjeta CRC es una clase, a pesar de que es claro que existen dos tipos de tarjetas CRC: tarjetas que representan entidades funcionales (lógica de negocios) y tarjetas que representan entidades no funcionales y transversales. Desde un enfoque más general, es válido aceptar que las tarjetas representan simplemente concerns. En principio esta premisa plantea dos cuestiones a resolver: i) un crosscutting concern puede ser identificado mediante las tarjetas CRC; y ii) un aspecto puede ser representado y modelado mediante una tarjeta CRC. Desde nuestra perspectiva ambas cuestiones son posibles.

Identificación de Crosscutting Concerns: Por naturaleza un crosscutting concern esta disperso y enmarañado por diferentes módulos de un sistema. Sin embargo los crosscutting concerns se caracterizan por tener un claro propósito y puntos de interacción regulares. Tomando por base estas propiedades, es correcto asumir que el "claro propósito" no es otra cosa que las responsabilidades (especificadas en la tarjeta) que se le ha asignado como entidad del sistema. Una tarjeta interactúa con otras a través de los servicios que ofrece. Entonces, el punto de interacción regular se establece cuando se define una colaboración. De esta forma, cuando una entidad representada por una tarjeta CRC colabora en diferentes tarjetas CRC, diseminándose y mezclándose, se estaría frente a la presencia de un "potencial crosscutting concern y posible aspecto temprano". Por ejemplo, en la

Figura 2 la tarjeta CRC A colabora con sus responsabilidades con las tarjetas C1, C2 y C3. La grafica muestra que la tarjeta A se ha diseminado en otras tarjetas, corresponde entonces analizar las responsabilidades de A para determinar si cumple con un requisito funcional o no funcional

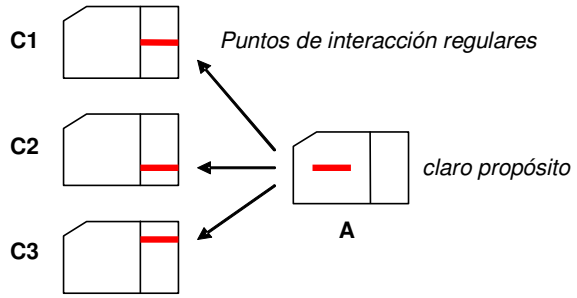

Figura 2: La tarjeta A colabora con las tarjetas C1, C2, C3.

Representación de Aspectos con Tarjetas CRC: En principio la representación de aspectos con Tarjetas CRC modifica parcialmente el esquema original del artefacto en dos sentidos: primero las clases no deberían contener información de aquellas entidades que han sido identificadas como crosscutting concern (principio de obliviousness [17]). Luego las entidades aspectos deben contener la referencia (implícitamente) de las entidades que entrecruzan (join-points). Esto requiere extender las tarjetas a una nueva dimensión, que se denomina "crosscut". La dimensión crosscut permite la composición de aspectos y clases. En la Figura 3 se representa la forma en que las tarjetas CRC Account y Logging son convertidas luego que se decide que Logging es un crosscutting concern y debe ser representado como aspecto.

En esta instancia de modelado conceptual, no se han considerado cuestiones referentes a la administración de conflictos entre aspectos (detección y resolución) como tampoco mecanismos de composición más específicos (after, before, etc.) o si las responsabilidades de los aspectos tempranos corresponden a avisos o introducciones. Esta decisión responde a dos razones: i) se entienden estas cuestiones mas como decisiones de diseño que refieren al espacio de solución que al espacio del problema que se pretende modelar; y ii) esencialmente el esquema CRC es en su naturaleza intrínseca un modelo simple, que se pretende preservar y podría verse oscurecido al incluir estas cuestiones, de manera anticipada. Sin embargo y sin perjuicio del resultado final estos aspectos pueden ser retomados y resueltos en una fase posterior de diseño detallado.

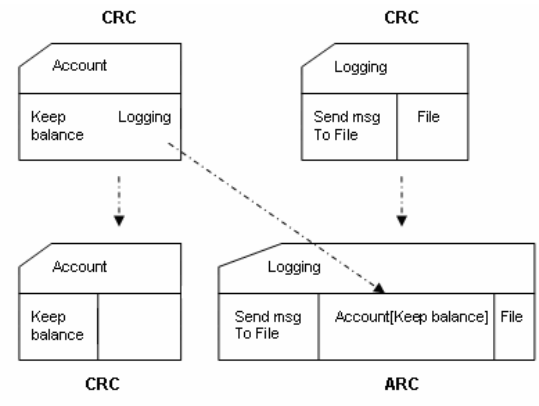

Figura 3: Tarjetas de Clase y de Aspecto

3.Identificación de Crosscutting Concerns

El proceso para identificar los aspectos tempranos a partir de las tarjetas CRC que se propone, establece 3 pasos básicos: 1) Identificación de Tarjetas CRC; 2) Clasificación de Tarjetas CRC en Subsistemas y 3) Cálculo y Análisis de Vistas. El gráfico de la Figura 4 ilustra los pasos descriptos.

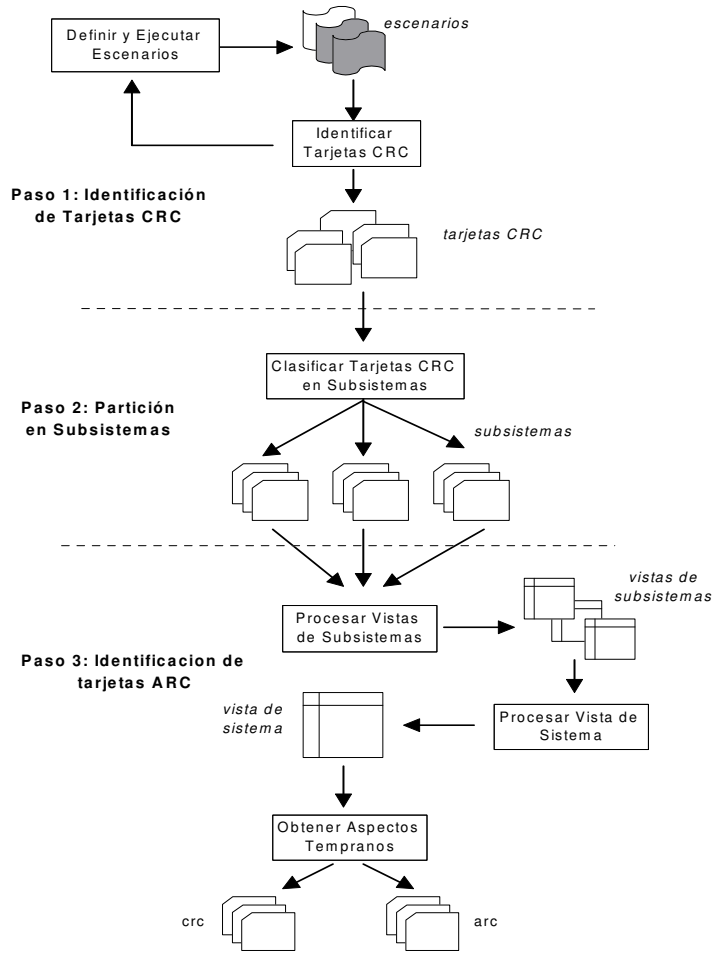

Figura 4: Pasos para identificar y modelar aspectos tempranos.

Paso 1: Los escenarios se plantean sin restricción según la técnica adoptada (tormenta de ideas, juego de roles, casos de uso, historias de usuario, etc.). La ejecución de cada escenario genera nuevas tarjetas CRC o la modificación de las existentes (al añadir nuevas responsabilidades y colaboraciones). Este proceso se itera hasta que se considere que la funcionalidad y requisitos del sistema están cumplidos. Esta fase produce como resultado: una colección de tarjetas CRC (preliminares) y un conjunto de escenarios.

Paso 2: En esta instancia las tarjetas CRC se clasifican en subsistemas. El concepto de subsistema que se aplica, retoma la visión de Wirfs-Brock [13] de subsistema "*Al descomponer la aplicación, inmediatamente se identifican las clases. Pero se deben encontrar otras entidades: piezas que tienen cierta integridad lógica, pero que son ellas mismas descompuestas en piezas más pequeñas. Estas piezas son los subsistemas. Un subsistema es un conjunto de clases colaborando para cumplir con un conjunto de responsabilidades. Aunque los subsistemas no existen durante la ejecución del software, son entidades conceptuales útiles*".

El resultado es un conjunto de tarjetas CRC (y escenarios) clasificados según sus propósitos en subsistemas.

Paso 3: Las vistas de subsistema y sistema constituyen el mecanismo que permite al desarrollador discernir y razonar sobre la naturaleza de los distintos concerns del sistema, representados con tarjetas CRC. El objetivo de las vistas es hacer emerger fácilmente aquellas tarjetas CRC que representan potenciales crosscutting concerns. Las vistas identifican a las tarjetas CRC que más se mezclan y dispersan por los subsistemas y por el sistema. Este paso produce una división entre tarjetas CRC y tarjetas ARC.

Una vista de subsistema (VS) es una matriz de NxM, donde N es la cantidad de tarjetas CRC del sistema y M es la cantidad de tarjetas CRC del subsistema que está en análisis. Sea S1 el subsistema que se está examinando y $t_{s11,}$ t_{s12}, t_{s13} el conjunto de tarjetas de CRC de S1, la VS(S1) se calcula mapeando todas las colaboraciones registradas en las tarjetas CRC de S1 en la fila correspondiente a la tarjeta CRC colaboradora. Si $VS[t_{si}][t_{sj}] = 0$ significa que t_{sj} no requiere de la colaboración de t_{si} para cumplir con sus responsabilidades y si $VS[t_{si}][t_{sj}] > 0$ significa que t_{sj} requiere la colaboración de t_{si} para cumplir con sus colaboraciones. La última columna de la vista representa la sumatoria de las colaboraciones de una tarjeta en dicho subsistema.

La vista de sistema (V) es una matriz similar a VS, pero las columnas representan a los subsistemas definidos y los valores de las celdas corresponden a la sumatoria calculada por cada fila de cada VS.

Las tablas 1 y 2 presentan dos ejemplos genéricos de vistas. Las tarjetas del sistema se han clasificado en 3 subsistemas, S1, S2 y S3 de la siguiente manera: S1 = $\{t_{s11,}$ t_{s12}, $t_{s13}\}$, S2=$\{t_{s21,}$ t_{s22}, $t_{s23}\}$ y S3 = $\{t_{s31,}$ t_{s32}, $t_{s33}\}$.

S1	t_{s11}	t_{s12}	t_{s13}	S&T
t_{s11}	0	0	0	0
t_{s12}	0	0	0	0
t_{s13}	0	0	0	0
t_{s21}	0	1	1	2
t_{s22}	0	1	2	3
t_{s31}	1	1	3	5
t_{s32}	0	0	0	0
t_{s33}	2	1	4	7

Tabla 1: Vista de Subsistema

	S1	S2	S3	S&T
t_{s11}	0	0	0	0
t_{s12}	0	0	0	0
t_{s13}	0	0	0	0
t_{s21}	2	1	1	4
t_{s21}	3	1	2	6
t_{s31}	5	1	3	9
t_{s32}	0	0	0	0
t_{s33}	7	1	4	12

Tabla 2: Vista de Sistema

Las vistas permiten identificar aquellas tarjetas que se dispersan y mezclan por el sistema. Sobre éstas el desarrollador deberá decidir si corresponde a un crosscutting concern. Luego que una tarjetas CRC se ha identificado como crosscutting concern y se decide representarla como aspecto, deben ser transformada afectando esta operación a las tarjetas CRC en las que presta colaboración.

4.TAOM

TAOM es un prototipo que en estas instancias permite experimentar de manera práctica nuestra propuesta. TAOM fue concebida como una herramienta de documentación pero que a la vez facilite la traceabilidad, composición, evolución y escalabilidad de los requerimientos. Las principales funcionalidades de TAOM son:

-Documentación
-Navegación Total
-Cálculo Automático de Vistas
-Conversión automática de tarjetas CRC a tarjetas ARC
-Representación XML del sistema.

Documentación: Esta funcionalidad permite registrar el modelado conceptual descripto. Las tarjetas y escenarios se registran sin restricciones. La definición de subsistemas consiste en una descripción del mismo y la asignación de Tarjetas CRC y escenarios al mismo.

Navegación total: Una de las propiedades principales de TAOM es que todas las abstracciones (tarjetas, escenarios, subsistemas) se enlazan con aquellas otras con las que mantiene algún tipo de relación. Desde un subsistema se puede navegar por las tarjetas (CRC-ARC) que lo definen, y desde cada tarjeta se puede navegar hacia sus colaboradores. De la misma manera, la navegación en las vistas es posible, accediendo a los detalles de los subsistemas, tarjetas, escenarios. Esta funcionalidad cumple con el requisito que la herramienta facilite indagar y explorar el modelo del sistema.

Cálculo de Vistas Automático: Esta funcionalidad genera automáticamente las vistas de subsistemas y de sistema.

Transformación Automática: Esta funcionalidad se refiere a la operación que convierte tarjetas CRC en tarjetas ARC, cuando una tarjeta CRC se ha identificado como crosscutting concern. Asimismo aquellas otras tarjetas CRC afectadas por las tarjetas ARC son actualizadas por el proceso. De esta forma se realiza automáticamente las transformaciones necesarias de esta a tarjeta ARC y de las tarjetas CRC relacionadas.

Exportación XML: Esta funcionalidad exporta toda la definición del sistema en formato XML. Esta operación facilita un posterior procesamiento mediante otras herramientas.

En la Figura 5 se visualizan dos de las operaciones mencionadas sobre el sistema ATM. La vista del subsistema Account y la transformación de tarjetas CRC en tarjetas ARC automática.

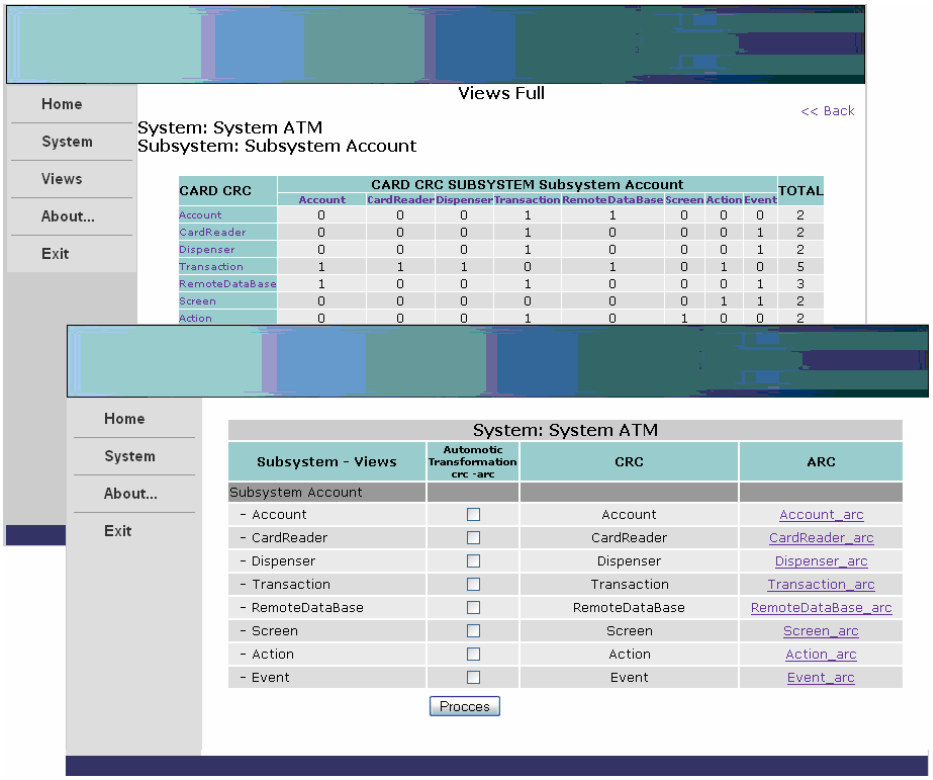

Figura 5: TAOM: Cálculo de Vista de un Subsistema y Transformación de tarjetas CRC a tarjetas ARC.

5.Traceabilidad, Composición, Evolución y Escalabilidad

A continuación se analiza la propuesta según los criterios de traceabilidad, composición, evolución y escalabilidad.

Traceabilidad: una tarjeta CRC será en diseño e implementación una clase y una tarjeta ARC será un aspecto. Los subsistemas serán paquetes. Asimismo una tarjeta CRC o ARC es el resultado de la ejecución de un escenario. Existe una correcta y directa correspondencia entre los distintos artefactos que facilitan la comprensión y mantenimiento de los requerimientos en sus distintas representaciones.

Composición: en principio la clave de la composición entre las tarjetas CRC y las tarjetas ARC esta dada en estas últimas, precisamente en la dimensión crosscut. Sin embargo en el esquema propuesto no se han definido mecanismos semánticos y sintácticos muy específicos. Esta actividad de especificación más detallada se deja para una etapa posterior de diseño, en la cual las tarjetas serán refinadas. También en esta etapa posterior se realizará el tratamiento de conflictos entre aspectos (detección y resolución).

Evolución: los cambios en los requerimientos implican que se agregan, modifican o eliminan entidades al sistema. Esta actualización puede ser costosa ya las que las tarjetas son artefactos físicos manuales. Pero con una herramienta como TAOM la evolución es factible y semi-automática.

Escalabilidad: un proyecto de gran complejidad será aquel en el cual existan decenas de tarjetas, la escalabilidad se manejará mediante la definición de subsistemas y el soporte de una herramienta como TAOM.

6.Trabajos Relacionados

Varios trabajos se han presentado con el objeto de identificar crosscutting concerns en etapas de ingeniería de requerimiento y/o modelado. La principal diferencia con ellos, es que estos se basan o proyectan en los artefactos provistos por la notación UML [18]. A continuación se mencionan algunos de ellos.

Constantinides [19] enfatiza la importancia de identificar y modelar crosscutting concerns desde las etapas iniciales del ciclo de vida del software. Este trabajo presenta un caso de estudio para investigar el modelado de crosscutting concerns principalmente en las actividades de análisis y diseño. Sin embargo aunque el autor propone adaptar las técnicas de análisis y diseño establecidas para un contexto orientado a aspectos, solamente describe como los crosscutting concerns pueden ser visualizados en diagramas de clases y secuencias, no sugiere técnicas para desarrollar estos artefactos.

Jacobson [20][21] aboga por la extensión de los casos de uso, tiene el mismo propósito que los aspectos en AOP y que dicho mecanismo puede ser utilizado en las actividades de requerimientos en AOSD.

Rashid [22][23] propone un proceso de modelado genérico para AORE, pero no explora los enlaces con las actividades análisis y diseño. Moreira [24] y Araujo [25] presentan un modelo simplificado para soportar un proceso general AORE (Ingeniería de Requerimientos Orientada a Aspectos) en [14]; estos trabajos componen requerimientos no-funcionales y funcionales usando extensiones de casos de uso y diagramas de secuencia.

Otra propuesta de extensión de UML para el diseño de aspectos fue presentada por Suzuki y Yamamoto [26]. Este trabajo extiende el metamodelo UML incluyendo una nueva clase de relación llamada *aspect*. Para modelar la relación aspect-class relationship, los autores abogan el uso de tipos

de relaciones de dependencia con estereotipos de realización, <<realize>>, ya provista por la notación UML. Sin embargo presentan una notación para declaraciones inter-type y no mencionan como los pointcuts o avisos pueden ser modelados. Además se enfocan en actividades de diseño y no exploran el enlace con actividades previas.

Theme/Doc [27] esta basado en la noción de theme (tema) que representa una característica de un sistema. A partir de requerimientos textuales se realiza la identificación de crosscutting themes. Theme/Doc provee vistas que asisten al desarrollador en determinar qué clase de relaciones existen entre los themes obtenidos desde los requerimientos, y si dichos themes son base o aspectos. Esta soportado por una herramienta que toma como entrada requerimientos textuales y palabras claves provistas por el desarrollador y las analiza léxicamente. Los resultados del análisis son presentados entonces en una vista gráfica conteniendo las acciones y conexiones entre ellas. Puesto que se asume que las acciones conectadas son una indicación de crosscutting y tangling, es entonces tarea del analista determinar cuál de esas conexiones indican realmente relaciones crosscutting.

Sampaio y otros [28], proveen un enfoque orientado a la exploración de aspectos desde documentos de requerimientos. La herramienta que desarrollaron puede trabajar con cualquier clase de documento textual independientemente de su estructura y automatiza parcialmente la identificación de concerns, view points y action words. La herramienta permite un desarrollo acelerado porque no impone una clase de formato específico y no depende del conocimiento previo del ingeniero de requerimientos sobre los requerimientos.

Lars Rosenhainer en [29] se enfoca en la identificación de requerimientos e influencias crosscutting en documentos de requerimientos ya existentes. Para Rosenhainer un requerimiento es una clase especial de concern. Los requerimientos que atraviesan transversalmente a otros son referidos como requerimientos crosscutting. La expresión influencia crosscutting es usada como un sinónimo para las relaciones entre dos requerimientos que está establecida por uno atravesando transversalmente (crosscutting) el otro. Sin embargo, no todas las dependencias de requerimientos son de naturaleza crosscutting.

Brito en [30] propone un enfoque cuyo principal objetivo es desarrollar un framework orientado a aspectos en el contexto de ingeniería de requerimientos. Para lograrlo, propone un modelo de ingeniería de requerimientos que se compone de varias tareas, y a su vez estas, se componen de subtareas. Estas tareas consisten en identificar concerns del sistema, a partir del uso de documentos y catálogos existentes (conteniendo terminología de requerimientos no funcionales), luego especificarlos, es decir, asignar responsabilidades, prioridades, etc., la siguiente tarea es modelar estos concerns en UML y la última es componer los concerns, que tiene que ver con identificar match points (abstracciones de join points en AspectJ), identificar crosscutting concerns y manejar conflictos.

7.Conclusiones

Las tarjetas CRC son una técnica simple pero efectiva que ha sido intensamente empleada para el modelado como diseño OO. De acuerdo a nuestra exploración, las tarjetas CRC pueden aplicarse para la identificación y modelado de los distintos tipos de concerns de un sistema. El método que se propone básicamente plantea analizar las tarjetas desde sus colaboraciones. Cuando una tarjeta CRC colabora en distintas tarjetas de distintos subsistemas se considera un potencial crosscutting concern y aspecto temprano. Las estrategias que se han planteado son simples: clasificar las tarjetas en subsistemas y calcular vistas. También se ha propuesto una nueva clase de tarjetas, para modelar específicamente los aspectos tempranos.

El trabajo actual y futuro esta centrado en completar el enfoque incorporando a la fase de diseño. El objetivo es redefinir las tarjetas de manera que abarquen los conceptos que el modelado conceptual

que no se han contemplado como ser: administración de conflictos entre aspectos; mecanismos de composición mas específicos para pointcuts, advices e inter-types.

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

Referencias

[1] AOSD 2002, 1st. International Conference on Aspect-Oriented Software Development. Gregor Kiczales, ed., (ACM Press, The Netherlands, 2002).

[2] Dijkstra E. (1976). "A Discipline of Programming", Prentice-Hall, 1.976.

[3] Hursch W., Lopes C. (1995). "Separation of Concern". Technical Report NU-CCS-95-03, Northeastern University, 1.995.

[4] Elrad, T., Filman, R. E., and Bader, A. Aspect-Oriented Programming: Introduction. Communications of the ACM, v. 44, n. 10, pp. 29-32. October 2001.

[5] Chitchyan R., Rashid A., Sawyer P., Garcia A., Pinto Alarcón M., Bakker J., Tekinerdogan B., Clarke S., Jackson A. "Survey of Aspect-Oriented Analysis and Design Approaches" AOSD-Europe-ULNC-9- 2005

[6] Lamsweerde A. (2001). "Goal-Oriented Requirements Engineering: A Guided Tour". 5th IEEE International Symposium on Requirements Engineering, 2.001.

[7] Jacobson I., Chirsterson M., Jonsson P., and Overgaard G. (1992). "Object-Oriented Software Engineering: A Use Case Driven Approach", 4 ed: Addison-Wesley, 1.992.

[8] Sommerville I., Sawyer P. (1997). "Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering". Annals of Software Engineering, vol. 3, pp. 101-130, 1.997.

[9] Rashid A., Sawyer P., Moreira A., and Araujo J. "Early aspects: A model for aspect oriented requirements engineering". In IEEE Joint Intl. Conference on Requirements Engineering, pages 199-202, Essen, Germany, September 2002. IEEE Computer Society Press.

[10] Bellin D. and Suchman Simone S. "The CRC Card Book" – Addison-Wesley 1997

[11] Wilkinson W. "Using CRC Cards: An Informal Approach to OO Development" – Cambridge University Press. 1996.

[12] Beck K.,Cunningham W. "A Laboratory For Teaching Object-Oriented Thinking", OOSPLA 1989, and special issue of SIGPLAN Notices vol. 24, num 10, 1989.

[13] Wirfs-Brock R., Wilkerson B., Wiener L. "Design Object-Oriented Software" Prentice Hall, 1990.

[14] Wirfs-Brock R. y Wilkerson B. "Object-Oriented Design: A responsibility –driven approach". OOSPLA 89, ACM-Conference on Object Oriented Programming, Systems, Languages and Applications, pages 71-75.

[15] Beck K. "Extreme Programming Explained: Embrace Change". Addison-Wesley – 1999.

[16] Cockburn A. "Using CRC cards" Informal draft of Humans and Technology Technical Memo HaT TR.99.01

[17] Filman, R., Friedman, D. "Aspect-oriented programming is quantification and obliviousness". Workshop on Advanced Separation of Concerns, Conference on Object-Oriented Programming, Systems, Languages and Applications. 2000.

[18] Booch G., Rumbaugh J., Jacobson I. "The Unified Modeling Language. User Guide". Addison-Wesley, 1999.

[19] Constantinides, C. “A case study on making the transition from functional to fine-grained decomposition”. ECOOP 2003 Workshop on Analysis of Aspect-Oriented Software (AAOS 03), Darmstadt.

[20] Jacobson, I. “Use Cases and Aspects – Working Seamlessly Together”, in Journal of Object Technology, vol. 2, no. 4, 2003, pp. 7-28.

[21] Jacobson, I. “Use Cases - Yesterday, Today, and Tomorrow”. The Rational Edge, 2003.

[22] Rashid, A.; Sawyer, P.; Moreira, A. and Araújo, J. “Early Aspects: a Model for Aspect-Oriented Requirements Engineering”, IEEE Joint Conference on Requirements Engineering, Essen, Germany, 2002.

[23] Rashid, A. Moreira, A. and Araujo, J. “Modularisation and Composition of Aspectual Requirements”. AOSD 2003, ACM, pp. 11-20.

[24] Moreira, A.; Araújo, J. and Brito, I. “Crosscutting Quality Attributes for Requirements Engineering”, 14th International Conference on Software Engineering and Knowledge Engineering, ACM Press, Italy, 2002.

[25] Araújo, J.; Moreira, A.; Brito, I. and Rashid, A. “Aspect-Oriented Requirements with UML”, Workshop: Aspect-oriented Modeling with UML, UML2002, Germany.

[26] Suzuki, J. and Yamamoto, Y. “Extending UML with Aspects: Aspect Support in the Design Phase”, In Proceedings of the 3rd Aspect-Oriented Programming (AOP) Workshop at ECOOP’99, Springer LNCS 1743. 1999.

[27] Baniassad and S. Clarke. Theme: “An approach for aspect oriented analysis and design”. International Conference on Software Engineering, 2004.

[28] Sampaio A., Loughran N., Rashid A. and Rayson P. “Mining Aspects in Requirements”. 2005, Chicago, Illinois, USA.

[29] Rosenhainer L. “Identifying Crosscutting concerns in Requirements Specifications”. Monday, October 25, 2004, Vancouver, Canada.

[30] Brito I. “Aspect-Oriented Requirements Engineering”. Trabajo de tesis de Isabel Brito, desarrollado en la Universidade Nova de Lisboa bajo la supervisión de la Doctora Ana Moreira.