Cuarto Congreso Colombiano de Computación 4CCC
Sociedad Colombiana de Computación
UNAB - UIS

Aspectos Tempranos: un enfoque basado en Tarjetas CRC

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

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
hreinaga@uarg.unpa.edu.ar

ABSTRACT
El Desarrollo de Software Orientado a Aspectos (AOSD) provee nuevas abstracciones para evitar la mezcla de distintos tipos de intereses y prevenir la diseminación de una funcionalidad en diversos módulos. Es deseable proveer técnicas de modelado, análisis y diseño orientadas a aspectos. Varios trabajos se han presentado con el objeto de identificar aspectos en etapas de ingeniería de requerimiento y/o modelado. En general estos métodos se basan o proyectan en los artefactos provistos por la notación UML. Esencialmente este trabajo explora la aplicación y extensión de las tarjetas CRC para la identificación, separación, representación y composición de aspectos tempranos. Además se presenta TAOM, una herramienta que da soporte al enfoque propuesto. Se propone un caso de estudio sencillo sobre el cual se aplica el enfoque.

Keywords
Programación Orientada a Aspectos, Tarjetas CRC, Aspectos tempranos, AOSD.

1. INTRODUCCION
El Desarrollo de Software Orientado a Aspectos (AOSD) [1] apunta a soportar la Separación de Concerns (asunto o competencia) [12] [14] con el objeto de modularizar aquellas funcionalidades que atraviesen transversalmente el sistema (crosscutting concerns). Como extensión de la Orientación a Objetos (OO), AOSD provee nuevas abstracciones para evitar la mezcla de distintos tipos de intereses y prevenir la diseminación de una funcionalidad en diversos módulos. Esta técnica de modularización permite encapsular la funcionalidad transversal en unidades llamadas aspectos. Como los intereses y funcionalidades de un sistema emergen en las etapas tempranas del proceso de desarrollo, es deseable proveer técnicas de modelado, análisis y diseño orientadas a aspectos. La importancia de identificar y modelar desde las etapas tempranas de desarrollo se ha plasmado en varios trabajos [9]. Las estrategias han apuntado a la extensión y adaptación de los métodos OO [15] [19] [27] los cuales dan soporte a la nueva abstracción [24].

Un enfoque menos explorado han sido las tarjetas CRC (Clase-Responsabilidad-Colaboración) [6] [29]. Las tarjetas CRC fueron presentadas por Beck y Cunningham en 1989 para la enseñanza de la Programación Orientada a Objetos (OOP) [5]. Desde entonces, esta técnica ha sido propuesta también para el modelado conceptual y/o diseño detallado de sistemas OO [4] [10] [31] [30]. La característica más sobresaliente de las tarjetas CRC es su simpleza y ductilidad, ya que una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema, a las cuales asigna responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre los stakeholders (participantes del proyecto), 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, atributos, relaciones de herencia, composición o dependencia.

Desde un enfoque puramente OO cada tarjeta CRC es una clase, pero 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 aspecto (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.

Esencialmente este trabajo explora la aplicación y extensión de este artefacto para la identificación, separación, representación y composición de aspectos tempranos. En la Sección 2 se presentan algunos trabajos relacionados; en la Sección 3 se expone el método propuesto; en la Sección 4 se presenta la extensión de tarjetas CRC a tarjetas ARC; en la Sección 5 se explica el manejo de conflictos entre aspectos con el enfoque propuesto; en la Sección 6 se presenta TAOM, una herramienta que da soporte al enfoque propuesto; en la Sección 7 se pone en practica el enfoque mediante el desarrollo de un caso de estudio; por último, en la Sección 8 presentamos las conclusiones.

2. TRABAJOS RELACIONADOS
Varios trabajos se han presentado con el objeto de identificar aspectos 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 [7]. A continuación se mencionan algunos de ellos.

Constantinides [11] 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 sugiriendo técnicas para desarrollar estos artefactos.

Jacobson [16] [17] aboga por la extensión de los casos de uso, tiene el mismo propósito que los aspectos en Programación Orientada a Aspectos (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 (Ingeniería de Requerimientos Orientada a Aspectos), pero no explora los enlaces con las actividades análisis y diseño. Moreira [20] y Araujo [2] presentan un modelo simplificado para soportar un proceso general AORE en [30]; 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 [28]. Este trabajo extiende el metamodelo UML incluyendo una nueva clase de relación llamada aspect. Para modelar la relación aspect-class 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 (puntos de corte) o avisos pueden ser modelados. Además se enfocan en actividades de diseño y no exploran el enlace con actividades previas.

Theme/Doc [3] 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 [26], 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 [25] 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 [8] 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 o puntos de unión en AspectJ), identificar crosscutting concerns y manejar conflictos.

3. IDENTIFICACIÓN DE ASPECTOS TEMPRANOS

Por naturaleza un crosscutting concern esta disperso y enmarañado por diferentes módulos de un sistema. Sin embargo este tipo de funcionalidad se caracteriza por tener un claro propósito y puntos de interacción regulares. Tomando por base estas propiedades, es correcto asumir que el "claro propósito" se corresponde con las responsabilidades que se asignan a una entidad del sistema especificada en una tarjeta CRC. A la vez, una entidad interactúa con otras a través de los servicios que ofrece, y se especifican a través de las colaboraciones. 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 aspecto temprano". Por ejemplo, en la Figura 1 la tarjeta CRC, RemoteDataBase colabora con sus responsabilidades con las tarjetas Account y Transaction. La grafica muestra que la tarjeta RemoteDataBase se ha diseminado en otras tarjetas, corresponde entonces analizar las responsabilidades de RemoteDataBase para determinar si cumple con un requisito funcional o no funcional.

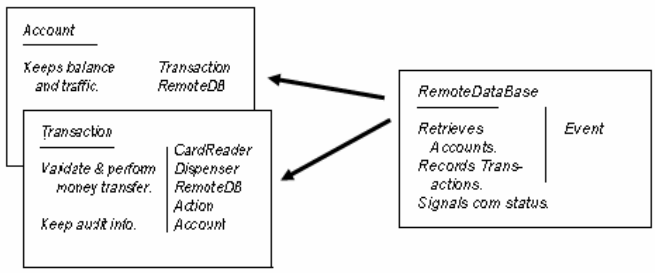

Figura 1: La tarjeta RemoteDataBase colabora con las tarjetas Account y Transaction.

3.1 Vistas de Subsistemas y Sistema

Cuando un sistema esta representado por un número considerable de tarjetas CRC, es recomendable que las tarjetas sean clasificadas o divididas en subsistemas [15]. Esta descomposición en subsistemas es una abstracción que permite manejar la complejidad de sistemas de mediano y gran tamaño. Una vista permite analizar el sistema desde sus partes (subsistemas) y desde el todo (sistema). El objetivo de las vistas es hacer emerger aquellas tarjetas CRC que representan potenciales aspectos tempranos, ya que ayudan a identificar a las tarjetas CRC que más se mezclan y dispersan por los subsistemas y por el sistema.

Las vistas se pueden calcular luego que todas las tarjetas CRC se han construido y se han clasificado en subsistemas. 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 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 responsabilidades. 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}}.

Tabla 1. Vista del Subsistema

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 2. Vista de Sistema

	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

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 tarjeta CRC se ha identificado como crosscutting concern y se decide representarla como aspecto, deben ser transformadas, afectando esta operación a las tarjetas CRC en las que presta colaboración.

4. TARJETAS ARC: MODELADO DE ASPECTOS TEMPRANOS

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 aspectos tempranos (principio de obliviousness [13]). Luego las entidades aspectos se representan en tarjetas denominadas ARC. Las tarjetas ARC definen responsabilidades, colaboraciones y cortes. En esta última dimensión se especifica explícitamente las entidades que se entrecruzan (join-points) y el tipo de composición (before-after-around) del corte transversal. La dimensión corte establece la composición de aspectos y clases.

En la Figura 2 a la izquierda se han dispuesto las tarjetas CRC originales denominadas Account, Transaction y RemoteDataBase. A la derecha, la transformación de las mismas luego que RemoteDataBase se considere un aspecto. Las colaboraciones de RemoteDataBase en las tarjetas Account y Transaction han desaparecido, en su lugar en la dimensión corte se ha especificado los join-points y métodos de composición, por ejemplo before[Account.Keep balance]. Así la responsabilidad Keep balance and traffic de la tarjeta Account, esta asociada a la responsabilidad Retrieves Accounts, la cual debe ser ejecutada antes.

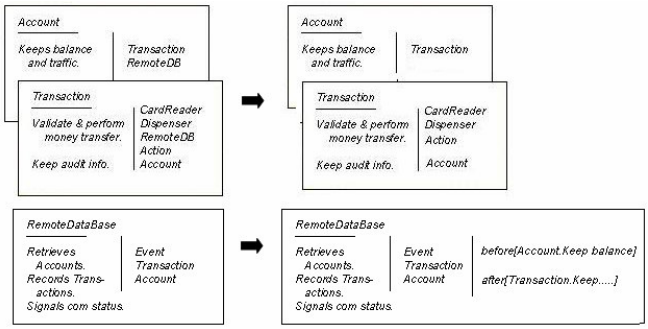

Figura 2: Tarjetas de Clase y de Aspect.

5. TRATAMIENTO DE CONFLICTOS

Otra cuestión que resulta conveniente controlar desde las etapas de modelado, es la que refiere al tratamiento de conflictos entre aspectos. Un conflicto ocurre cuando dos o más aspectos compiten por su activación [21]. En el enfoque presentado, dos o más tarjetas ARC están en conflicto cuando han definido el mismo join-point y método de composición en la dimensión corte. En ciertos casos los conflictos entre aspectos deben ser resueltos ya que la ejecución sin control de los mismos puede provocar comportamientos anómalos. En este sentido, aunque la identificación de conflictos entre aspectos puede realizarse en forma manual o automática, la resolución esta absolutamente condicionada a los mecanismos provistos por el lenguaje de aspectos empleado en la implementación. El método de resolución ampliamente soportado se refiere al establecimiento de precedencias, orden o prioridad entre aspectos. Otros métodos de resolución más flexibles que incluyan mecanismos de exclusión y/o condicionales, no son soportados en las herramientas de mayor madurez y consolidación, como AspectJ [18]. La resolución de conflictos implica definir métodos o reglas de

composición de aspectos que deben operar al momento de la composición del sistema.

Siguiendo el espíritu del enfoque, se propone para la administración de conflictos entre aspectos tempranos, una tarjeta especial, denominada "Conflicts", que tiene por objeto representar los conflictos detectados y especificar un método de resolución. En la Figura 3 se presenta un ejemplo. Cada fila representa un conflicto. La primera columna identifica el join-point objeto de la superposición, en la segunda columna se indican los aspectos en conflicto y en la tercera columna el método de resolución. La construcción de la tarjeta manualmente, implica transcribir la dimensión corte de las tarjetas ARC una única vez, y luego indicar en la segunda columna, los aspectos que la contienen en la dimensión corte. Luego se eliminan las filas que no especifiquen más de un aspecto en la segunda columna. Finalmente, por cada conflicto se establece el método de resolución (tercera columna).

Conflicts

Account.Keep Balance	*RemoteDataBase* *Logging*	*1.Logging* *2.RemoteDataBase*
Screen.Display Prompts	*Event* *Action*	*1.Event* *2.Action*

Figura 3: Tarjeta de Conflictos.

La especificación de los conflictos y su resolución en un mismo y único artefacto responde a facilitar futuras decisiones de implementación. La resolución de un conflicto puede tener incidencias sobre otros por eso una documentación unificada puede prevenir errores. Por ejemplo, si un sistema se va a implementar en AspectJ[18] y en el mismo existen dos aspectos que plantean dos conflictos diferentes (es decir sobre join-points, pointcuts, advice diferentes) la resolución que se establezca sobre uno define la resolución del otro. El examen y análisis unificado de todos los conflictos, permitirá mas fácilmente identificar este tipo de situaciones y decidir lo que corresponde.

6. TAOM

TAOM es una herramienta que da soporte automático al enfoque propuesto. La Figura 4 ilustra un esquema de la funcionalidad provista por TAOM y como debe ser operada. Se distinguen 3 funcionalidades importantes:

(1) Las tarjetas CRC obtenidas de las sesiones de los stakeholder son registradas en TAOM. En este paso también se almacenen los escenarios y sus relaciones con las tarjetas, y la división del conjunto de tarjetas en subsistemas;

(2) TAOM genera automáticamente todas las vistas, lo cual permite al desarrollador identificar los aspectos tempranos. Este selecciona que tarjetas CRC serán aspectos y TAOM automáticamente transforma las tarjetas CRC en ARC y realiza las modificaciones sobre las restantes CRC que correspondan;

(3) Luego que las tarjetas ARC (aspectos) están definidas, TAOM identifica automáticamente los conflictos y permite especificar la resolución de cada uno de estos en la tarjeta Conflicts. De manera automática, los conflictos se pueden visualizar y filtrar por diversos criterios (join-points o aspectos).

Todo el modelado del sistema definido a través de las tarjetas CRC, tarjetas ARC y la tarjeta Conflicts, son exportables mediante la generación de ficheros XML, lo que facilitaría posteriores procesamientos.

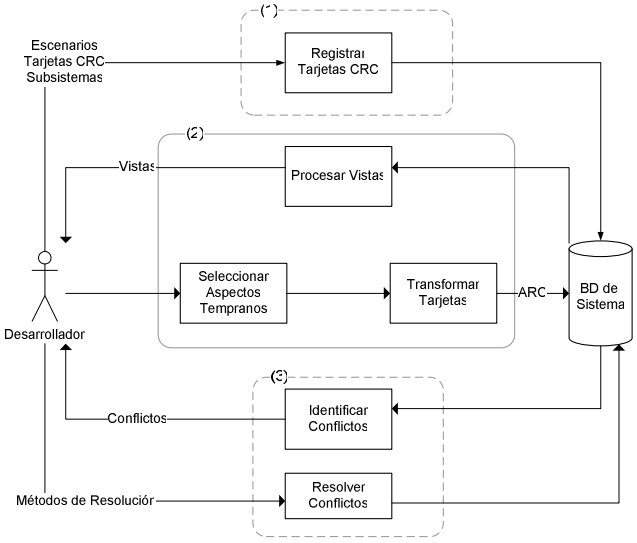

Figura 4: Esquema Funcional de TAOM.

TAOM es una herramienta sencilla pero muy útil para los desarrolladores, además de documentar la información del sistema sin complicaciones. La información se ingresa a través de formularios que siempre resultan más fáciles de construir que los gráficos para el desarrollador. Pero además TAOM es una herramienta que favorece la evolución y escalabilidad de los sistemas. Los cambios en los requerimientos implican que se agregan, modifican o eliminan entidades al sistema. Esta actualización puede ser costosa ya que las tarjetas son artefactos físicos manuales. Pero con una herramienta como TAOM, la evolución es factible y semi-automática porque las relaciones y asociaciones entre todas las tarjetas están almacenadas. En proyectos de mediana o gran envergadura existirán decenas de tarjetas, la escalabilidad se manejará mediante la definición de subsistemas y el soporte de una herramienta como TAOM.

7. CASO DE ESTUDIO

A continuación se propone un breve caso de estudio en el cual se ha aplicado el enfoque propuesto.

Una universidad va a implementar la gestión de trámites del Departamento de Alumnos y Estudios mediante un servicio web con el objeto de hacer más eficiente la atención de los alumnos y evitar los cuellos de botellas y largas colas que se generan en las ventanillas de atención al público. Los cuatro trámites que podrán ser canalizados a través del sistema son: certificado de alumno regular (CAR); permiso de examen (PE); certificado analítico (CA); e inscripción para cursar asignatura (ICA). El sistema debe en cada caso, procesar la solicitud y enviar un mail al alumno que comunique si el trámite se ha cumplimentado o no.

El procesamiento de las solicitudes varía según el trámite al que refiere, por ejemplo el certificado de alumno regular implica que el alumno haya aprobado al menos dos exámenes finales en los últimos 12 meses; el permiso de examen requiere que las

asignaturas correlativas estén cursadas y/o aprobadas; el certificado analítico es una constancia de asignaturas aprobadas; y la inscripción a asignatura, requiere que las asignaturas correlativas estén cursadas. Todos estos procesos se realizan mediante operaciones de consulta a la BD (base de datos) de alumnos. Para todos los casos sólo se procesan trámites de alumnos registrados en la BD, para lo cual se toma el DNI (Documento Nacional de Identidad) como dato de autentificación.

En la Figura 5 se pueden observar las 6 tarjetas CRC obtenidas de esta breve narrativa. Cada uno de los trámites se ha representado como una entidad del sistema. Asimismo cada uno tiene los datos que requiere para su posterior procesamiento. Por ejemplo, para realizar la inscripción a una asignatura, solo se requiere el DNI del alumno, el código de la asignatura a la que se va inscribir y el código de la carrera correspondiente. Luego el DNI será utilizado para que otra entidad denominada Autentificar determine si es un alumno de la universidad. Procesar Inscripción, realiza las operaciones necesarias para efectuar la inscripción particular. Por último, la operación Notificar envía un mail al alumno para notificar el resultado del trámite, esta operación se realiza con la colaboración de la entidad Mail.

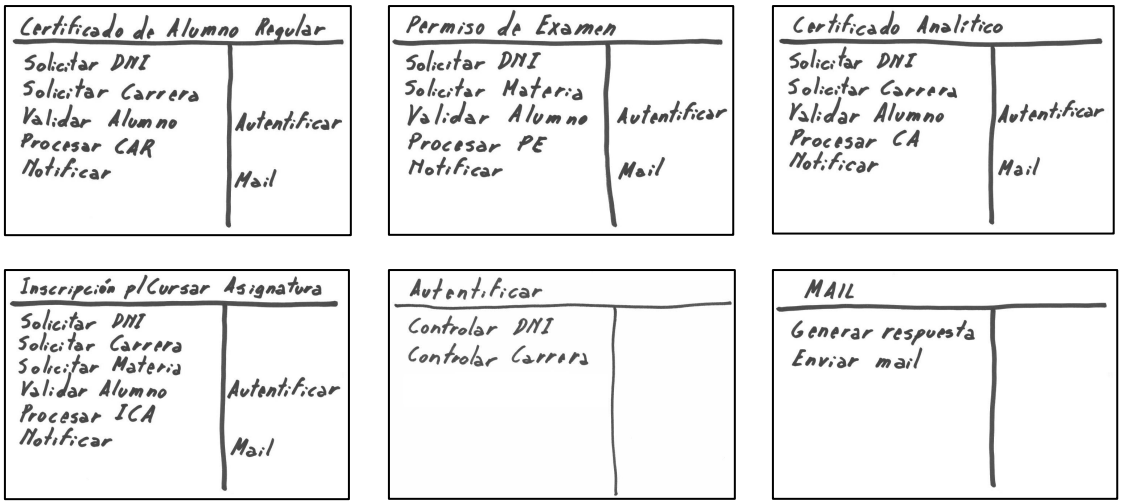

Figura 5: Tarjetas CRC del caso de estudio.

Las 6 tarjetas CRC son registradas en TAOM, que calcula automáticamente la vista que se muestra en la Figura 6. En este caso resulta que Autentificar y Mail atraviesan los distintos trámites, para colaborar con servicios no funcionales referidos a seguridad y notificación.

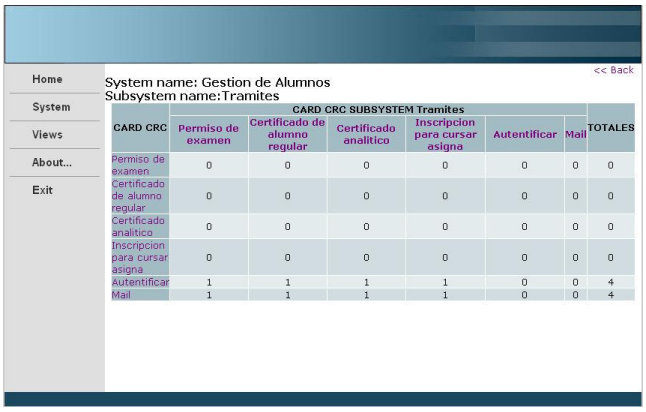

Figura 6: TAOM: Cálculo de Vista de un Subsistema.

Las entidades Autentificar y Mail, se representan como aspectos mediante tarjetas ARC (Figura 7 y 8).

Autentificar		
Controlar DNI **Controlar Carrera**		**before[Permiso de Examen.Validar Alumno]** **before[Certificado de Alumno Regular.Validar Alumno]** **before[Certificado Analitico.Validar Alumno]** **before[Inscripción para cursar Asignatura.Validar Alumno]**

Figura 7: Tarjeta ARC de Autentificar.

Mail		
Generar respuesta **Enviar mail**		**after[Permiso de Examen.Notificar]** **after[Certificado de Alumno Regular.Notificar]** **after[Certificado Analitico.Notificar]** **after[Inscripción para cursar Asignatura.Notificar]**

Figura 8: Tarjeta ARC de Mail.

En el caso descripto no existen conflictos entre los aspectos Autentificar y Mail, ya que aunque éstos cortan transversalmente las mismas tarjetas CRC, los métodos de composición son diferentes en cada caso.

8. CONCLUSIONES

Las tarjetas CRC son en su esencia un modelo simple, fácil de comprender y aplicar. Esta propuesta ha tratado de mantener en principio estas premisas, al plantear su uso y extensión para la identificación, modelado, representación y composición de aspectos.

Mediante el análisis de las colaboraciones se pueden identificar aspectos tempranos. La definición de un nuevo tipo de tarjeta, como la tarjera ARC, permite modelar y representar una entidad como aspecto, para lo cual se ha añadido una dimensión corte que posibilita establecer los mecanismos de composición. Asimismo se ha considerado la ocurrencia de conflictos entre tarjetas ARC, cuyo tratamiento se delega en una nueva clase de tarjeta que representa los conflictos identificados y su resolución.

Aunque la propuesta debe ser más probada y refinada, podría resultar válida especialmente para aquellos proyectos de software que se conciban desde una mirada ágil de desarrollo. A su vez, también pueden ser empleadas en forma complementarias con métodos más pesados basados en UML.

REFERENCIAS

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

[2] Araújo, J., Moreira, A., Brito, I. and Rashid, A. 2002. "Aspect-Oriented Requirements with UML", Workshop: Aspect-oriented Modeling with UML, UML2002, Germany.

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

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

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

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

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

[8] 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.

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

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

[11] Constantinides, C. 2003. "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.

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

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

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

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

[16] Jacobson, I. 2003. "Use Cases - Yesterday, Today, and Tomorrow". The Rational Edge.

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

[18] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W. 2001. "An Overview of AspectJ". ECOOP.

[19] Lamsweerde A. 2001. "Goal-Oriented Requirements Engineering: A Guided Tour". 5th IEEE International Symposium on Requirements Engineering.

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

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

[22] Rashid A., Sawyer P., Moreira A., and Araujo J. 2002. "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.

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

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

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

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

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

[28]Suzuki, J. and Yamamoto, Y. 1999. "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.

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

[30] Wirfs-Brock R., Wilkerson B. 1989. “Object-Oriented Design: A responsibility –driven approach”. OOSPLA 89, ACM-Conference on Object Oriented Programming, Systems, Languages and Applications, pages 71-75.

[31] Wirfs-Brock R., Wilkerson B. y Wiener L. 1990. “Design Object-Oriented Software” Prentice Hall.

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