Adaptaciones de workflows para administración de seguridad mediante MDD y Aspectos

TESIS DE MAESTRÍA

Fáber Danilo Giraldo Velásquez

Departamento de Informática y Sistemas
Escuela de Ingeniería
Universidad EAFIT

Junio de 2011

Documento elaborado en LaTeXusando la plantilla TeXiS v.1.0+.
Mayor información sobre TeXiS en
http://gaia.fdi.ucm.es/grupo/projects/texis/

Este documento está preparado para ser impreso a doble cara.

Adaptaciones de workflows para administración de seguridad mediante MDD y Aspectos

Informe presentado para optar al título de
Mágister en Ingeniería con énfasis en Informática
Línea de profundización en Ingeniería de Software

Supervisado por:
Raquel Anaya de Páez PhD - Universidad EAFIT.
Mireille-Blay Fornarino, PhD - University of Nice-Sophia Antipolis

Departamento de Informática y Sistemas
Escuela de Ingeniería
Universidad EAFIT

Junio de 2011

A L. D. Muñoz
y
a la familia Giraldo Velásquez

Agradecimientos

Vaya! Creo que esta es la primera vez en la que estoy en la obligación de expresar tantos agradecimientos al mismo tiempo, así que haré lo posible por considerar en estos renglones a todos quienes con su maravillosa amistad y generosidad han permitido culminar exitosamente este trabajo.

En primer lugar, expreso mis agradecimientos al ingeniero *Helmuth Trefftz*, director del Departamento de Informática y Sistemas, y al ingeniero *Alberto Rodríguez*, decano de la escuela de Ingeniería de la Universidad EAFIT, por liderar la extensión del programa de maestría en Ingeniería hacia la Universidad del Quindío, compartiendo el conocimiento y la dinámica de una institución de tanto prestigio como EAFIT, con mentes activas e inquietas que soportan la dinámica informática en el departamento del Quindío. Ha sido y seguirá siendo un honor trabajar al lado de personas supremamente amables, humildes, visionarias y pujantes como ustedes! ustedes son el fiel reflejo del empuje y tenacidad paisa!

Extiendo también mis agradecimientos a la doctora *Raquel Anaya de Páez*, directora del grupo de Ingeniería del Software en la Universidad EAFIT, quien desde el primer momento mostró su amable colaboración y disposición para orientar este trabajo. Ella, conjuntamente con el profesor *Luís Fernando Londoño* (a quien también presento agradecimientos) me dieron la oportunidad de conocer el mundo de los procesos de negocio, worflows, las arquitecturas empresariales, el paradigma orientado a aspectos, y tópicos avanzados para el desarrollo de software, mediante los espacios académicos impartidos en el programa de maestría.

Una buena parte de esta sección está dedicada a dos personas que, aunque nunca he tenido la oportunidad de conocer personalmente, jugaron un papel importante en la ejecución de este trabajo, puesto que compartieron sus

valiosos conocimientos y dedicaron muchas horas para brindar asesoría en forma remota usando email y *skype* como mecanismo de interacción; ellos son: la doctora *Mireille Blay-Fornarino* de la Universidad de Nice-Sophia Antipolis, y el doctor *Sébastien Mosser* de la Universidad de Lille I. Desde Agosto de 2009 la doctora Blay-Fornarino inició una comunicación permanente vía email, permitiéndome conocer nuevas tendencias sobre ingeniería de modelos, workflows y aplicación de principios de aspectos, pasando por proyectos como *Moteur*, **ADORE** y *Neurolog*. El carisma y dedicación de la doctora Blay-Fornarino se refleja en los mensajes intercambiados; cada uno de estos mensajes incluían asesorías valiosas y escenarios de investigación a ser contemplados en el desarrollo de la propuesta. A la doctora Blay-Fornarino le agradezco profundamente por toda su dedicación y ayuda en sus asesorías vía email y skype. Al doctor Mosser (autor del método **ADORE**) le agradezco por toda su ayuda en la solución de problemas técnicos derivados del trabajo con **ADORE**. Aprovecho la oportunidad para agradecer al doctor *Philippe Collet* (Universidad de Nice-Sophia Antipolis) por haber sido el contacto inicial de interacción con el grupo de investigación *MODALIS* de dicha institución.

Agradezco también a los profesores *Sergio Ochoa* y *Alexander Bergel* por su valiosa colaboración recibida durante la estancia realizada en Junio de 2010 en el Departamento de Ciencias de la Computación de la Universidad de Chile. Las instrucciones recibidas durante dicha estancia permitieron formular la propuesta de investigación relacionada en este documento.

Finalmente agradezco a todos los compañeros del grupo de investigación *SINFOCI* de la Universidad del Quindío (*William Joseph, María Lilí, Alexandra, Jaime, Leonardo, Hamilton, Jorge Iván*), por sus valiosos aportes y permanente motivación y ayuda para desarrollar este trabajo.

Resumen

Este documento presenta una propuesta para abordar el problema de adaptación de workflows, mediante la incorporación de propiedades de seguridad a nivel de modelado de proceso de negocio, controlando el impacto de dicha incorporación sobre el proceso que se interviene y la gestión de intereses transversales (*concerns*) que se entrelazan unos con otros. La pro-puesta define cómo administrar en forma mantenible, reutilizable y extensible los intereses o concerns derivados de las propiedades de seguridad, a nivel de modelos, de acuerdo con el contexto, las variaciones de las normas y requisitos de proceso de negocio, y estándares de seguridad propuestos para entornos de servicios distribuidos.

Generalmente las operaciones de seguridad que hacen parte de procesos de negocio (como el control de acceso) son implementadas como modificaciones del proceso en sí mismo, consecuentemente, un proceso de negocio que no contiene operaciones de seguridad se modifica para hacer frente a este tipo de operaciones (autenticación, control de acceso, firmas digitales, certificados, y cualquier otra operación relacionada con la protección de la información). Las prácticas de modelado e implementación de workflows de procesos de negocio, de acuerdo con paradigmas basados en *SOA* y *separación de intereses*, promueven la definición y estandarización de servicios que sean invocados/reutilizados en múltiples contextos. Por tal motivo se evidencia que los estándares de seguridad son mapeados o soportados desde niveles conceptuales para que éstos sean considerados en las fases tempranas de construcción de software o procesos de negocio. La conceptualización de las operaciones de seguridad permite que este atributo de calidad sea considerado como un comportamiento transversal compuesto por servicios soportados sobre infraestructuras subyacentes específicas. Este trabajo usa el estándar *XACML* de control de acceso, relacionando los servicios definidos por éste para ejecutar la validación de accesos sobre diferentes recursos; estos

servicios se convierten en los puntos de entrelazado cuando son incorporados dentro de un proceso de negocio existente a través de una unidad de modelado aspectual.

La propuesta bajo consideración hace uso del método **ADORE** con el fin de modelar los procesos de negocio como unidades aspectuales que posteriormente se entrelazan para formar composiciones y orquestaciones de servicios. Dado el enfoque de representación de comportamientos planteado por **ADORE**, en donde cada unidad aspectual contiene un comportamiento particular independiente de su contexto de origen (funcional o no funcional), se propone el enriquecimiento de dicho método mediante la incoporación de técnicas para modelados de intereses transversales basadas en *Theme/UML* en aras de de definir unidades aspectuales que representan comportamientos derivados desde atributos de calidad y estándares que implementan dichos atributos. La meta de la capa de abstracción incorporada es permitir la definición de comportamientos génericos reusables en un conjunto de procesos de negocio. El método **ADORE** también es enriquecido a nivel conceptual con la integración de elementos formulados en metamodelos que consideran características específicas de seguridad basada en control de acceso bajo el modelo *RBAC* y el estándar *XACML*. La integración de elementos a nivel de metamodelado permite justificar conceptualmente el modelado de unidades aspectuales encargadas de manejar comportamientos concretos de seguridad. La integración presentada soporta las modificaciones sobre las bases conceptuales de **ADORE** para definir unidades responsables de gestionar y mantener atributos de calidad direccionados por comportamientos (invocaciones de servicios subyacentes).

En este trabajo se expone un ejemplo de un proceso de negocio donde se abordan intereses definidos desde una especificación de requisitos. Los workflows de procesos de negocio son enriquecidos con la adición de unidades que contienen operaciones de seguridad (control de acceso basado en roles y encripción). Finalmente, para ilustrar el contexto del workflow asociado al caso de estudio seleccionado, así como la complejidad y ubicación de las unidades aspectuales de seguridad definidas en este trabajo, se aplica una técnica de visualización de procesos de negocio que permite exponer la complejidad de las unidades aspectuales de negocio y las unidades de seguridad con sus respectivas incorporaciones dentro de las composiciones y orquestaciones del proceso de negocio.

Abstract

This document present a proposal for addressing the problem of workflows adaptation, through the inclusion of security properties at a business modeling level, controlling its impact on the business processes that are intervened, and the management of crosscutting concerns that are woven with each other. The proposal defines how concerns derived from security properties can be managed in a maintainable, reusable and extensible way at model level, according to the context, variations of rules and requirements of business processes, and security standars formulated by distributed services enviroments.

Generally, security operations that are part of business processes (such as access control) are implemented as modifications of the business processes themselves; consequently, a non-secure business process is modified to deal with specific secure operations (authentication, access control, digital signatures, certifications, and any operation related with protection of information). Practices of modeling and development of business process workflows, according with *SOA* and *separation of concerns* paradigms, promotes the de-finition and standarization of services; it can be invoked/reused in multiples contexts. Therefore, it is evident that security standards are mapped or supported since conceptual levels with the purpose of these services can be considered in early stages of software or business process development. Conceptualization of security operations allows this quality attribute that can be considered as a crosscutting concern composed by services supported over specific underlying infraestructures. This work uses the *XACML* access control standard, relating the services defined by the standard for performing the validation of access over several resources; these services are converted in the woven points when it is incorporated into a existing business process through a aspectual model unit.

The proposal under consideration use the **ADORE** method for modeling the business process as aspectual units that later are woven for forming compositions and orchestrations of services. Due to focus of behaviours representation exposed in **ADORE**, where each aspectual unit contains a particular behavior independent of its origin context (functional or non-functional), we propose the enrichment of this method by the incorporation of approaches for crosscutting concerns modeling based on *Theme/UML* in order to define aspectual units that represent behaviours derived from quality attributes and standards that implement these attributes. The goal of a layer abstraction added is to allow the definition of generic reusable behaviours in a set of business processes. Also, **ADORE** is enriched at a conceptual level with the integration of elements at metamodel level that consider specific features of access control security under the *RBAC* model and the *XACML* standard. The integration of elements at metamodel level helps to justify conceptually the modeling of aspectual units resposibles of managing concrete security behaviours. The integration exposed serves to support modifications over the conceptual basis of **ADORE**, for defining units resposibles of managing and maintaning quality attributes driven by behaviours (underlying services invocations).

This work exposed an example of a business process where concerns derived from a requirement specification are managed. The business process workflows are enriched with the addition of units that contain security operations (access control based on roles and encryption).

Finally, in order to show the context of the workflow associated with the study case chosen, as well as the complexity and placement of aspectual units defined in this work, we apply a business process visualization approach for exposing the complexity of aspectual business units and security units with its respective incorporations into the compositions and orchestrations of business process.

Índice

Índice de figuras

Capítulo 1

Introducción

La adaptación de workflows que soportan procesos de negocio se fundamenta en la reutilización de servicios y procesos, generalmente basadas en especificaciones funcionales que deben ser aplicadas sobre los procesos de negocio. La incorporación de requisitos no funcionales en un workflow tradicionalmente contiene un elevado nivel de supervisión y control, debido a que debe preservarse la funcionalidad del proceso de negocio, a la vez que se abordan dominios específicos con diferentes expertos (dominios legales, comerciales, contexto del workflow en sí mismo manejado por requisitos no funcionales) . Muchas de las técnicas para adaptación de workflows se basan en procedimientos manuales (cambios en archivos de configuración, edición de políticas, introducción de código especíco, entre otros) ejecutados por el diseñador del workflow, quien conoce donde se necesita adaptar el workflow[56]. Esto implica que, en aras de enfrentar los requisitos claves no funcionales, se tiene que dise-ñar workflows acordes con su contexto, invirtiendo tiempo en actividades no orientadas al negocio. Incluso, si las infraestructuras de producción se mejoran en términos de tamaño, cantidad de datos manipulados y cantidad de cálculos computacionales ejecutados, el diseñador del workflow todavía debe asegurarse de que los workflows se ajusten a los requisitos iníciales, y también debe cerciorar que los componentes para administración de la seguridad se han conectado correctamente al workflow.

Actualmente en el diseño y especificación arquitectónica de sistemas software basados en servicios se promueve la incorporación de atributos de calidad o requisitos no funcionales utilizando técnicas de desarrollo de software orientado a Aspectos (AOSD). El proyecto presentando a consideración pretende explorar cómo el uso de técnicas basadas en aspectos puede aportar a

la definición de intereses transversales (*croscutting corcens*) de requisitos no funcionales particularmente relacionados con la seguridad, que puedan ser aplicados a la adaptación de workflows de procesos de negocio.

Este documento presenta una propuesta de investigación dirigida a explorar cómo se puede lograr la adaptación de workflows de procesos de negocio, a nivel de modelado, considerando especificaciones concretas relacionadas con el atributo de calidad de la *seguridad*, y cómo esta especificación puede ser articulada en un framework ya existente.

La seguridad es uno de los intereses transversales más importantes a ser abordadas en el diseño y modelado de un workflow, y debe ser considerada desde etapas tempranas de la especificación de los workflows. Particularmente este trabajo se centrará en operaciones de seguridad basadas en *control de acceso* bajo el modelo *RBAC* y el estándar *XACML* (ver sección 4.1). Se pretende validar si el uso de técnicas basadas en ingeniería de modelos, modelado orientado a aspectos, y uso de frameworks para orquestación de servicios, contribuyen a la definición de adaptaciones sobre workflows de procesos de negocio, derivadas éstas desde operaciones propias del dominio de la seguridad.

El desarrollo de la idea de investigación se presenta en los capítulos a continuación relacionadas: el capítulo II presenta los fundamentos conceptuales para la adaptación de workflows de procesos de negocio, involucrando el principio de separación de intereses (*Separation of Concerns - SoC*), desarrollo orientado a aspectos, desarrollo basado en modelos y adaptaciones de workflows basadas en seguridad; el capítulo III presenta el método **ADORE**, el cual se usa como plataforma base para el planteamiento de las adaptaciones de workflows. El capítulo IV introduce el dominio de *seguridad basada en control de acceso* usando el modelo *RBAC* y el estándar *XACML*; dicho dominio establece consideraciones conceptuales (expuestas a nivel de modelado) para la adaptación a aplicar sobre un workflow de procesos de negocio que requiera incorporar comportamientos de seguridad. El capítulo V presenta la propuesta de integración del estándar *XACML* en el método **ADORE**. El capítulo VI expone una técnica para visualizar la complejidad del workflow con respecto a las adaptaciones de seguridad incorporadas. Finalmente, el capítulo VII presenta las conclusiones del trabajo realizado y los trabajos futuros derivados a partir de esta investigación.

1.1. Palabras Clave

- Workflows
- Procesos de Negocio
- Aspectos
- MDD
- Seguridad
- RBAC
- XACML
- ADORE
- Adaptaciones

1.2. Planteamiento del problema

De acuerdo con la *Workflow Management Coalition*[1], un **workflow** es la automatización de un proceso de negocio, de forma total o parcial, durante la cual los documentos, información o tareas son comunicados de un partici-pante a otro para cumplir con una acción acorde a un conjunto de reglas procedimentales[2]. Un workflow es una herramienta software para definir, administrar y representar procesos de negocio complejos. Los workflows reflejan aspectos organizacionales de procesos de negocio, como la estructura, sincronización y ordenamiento (flujo) de tareas e información [32].

Un ejemplo típico de soporte a procesos de negocio se puede encontrar en las infraestructuras de producción para respaldar investigaciones médicas en diversas necesidades, como *Health-e-Child* [3], el proyecto *NeuGrid3* [4], entre otras. Este tipo de infraestructuras ofrecen automatización de procesos que se ejecutan sobre una grid, donde estos procesos consideran un importante volumen de datos a computar. El desarrollo de aplicaciones basadas en Grid

[1] http://www.wfmc.org/

[2] *Workflow Management Coalition Terminology & Glossary*, Document Number WFMCTC-1011.

[3] http://www.health-e-child.org/

[4] http://www.neurogrid.net/

computing para este tipo de aplicaciones consiste en *(i)* la definición de un **workflow** para el análisis de datos y/o servicios (lo cual representa el dominio de los procesos de negocio, como el caso de los datos médicos para los proyectos referenciados anteriormente), mientras *(ii)* la implementación de restricciones transversales para responder a la necesidad de ajustarse a restricciones legales en materia de protección de datos y seguridad.

Es evidente que los workflows deben adaptarse a los cambios legales, acuerdos comerciales, y modificaciones del contexto. Algunas de estas adaptaciones son estáticas que van siendo integradas en nuevas versiones del producto, al diseñarse incrementalmente nuevos workflows basados en los ya existentes. Otras adaptaciones deberían realizarse de forma dinámica en tiempo de ejecución para responder a fallas o reducción en la calidad del servicio. Debido a diferentes consideraciones (costos, eficiencia y seguridad), varias adaptaciones pueden ocurrir en un mismo workflow. Por lo tanto el usuario debe asegurarse nuevamente que el workflow resultante continúe siendo *válido*.

Los cambios en ambientes y contextos de negocio han generado la necesidad de producir tecnología y herramientas para soportar la administración eficiente y efectiva de workflows de procesos tanto a nivel de negocio como a nivel científico. Como resultado existen numerosos intentos para mejorar sistemas de información a través de funciones avanzadas de administración de procesos, que van más allá de la simple manipulación de tareas independientes [4].

Con base en el trabajo presentado en [75] es posible deducir los retos generados por la adaptación de los workflows, como son:

- Mejorar los procesos de misión crítica de una organización.
- Aumentar la flexibilidad de aplicaciones de negocio.
- Administrar los cambios del workflow altamente acoplados a aspectos no funcionales del contexto del workflow y/o nuevas versiones de los workflows.
- Manejar múltiples perspectivas del workflow mediante la incorporación de esquemas o especificaciones de alto nivel.
- Garantizar la efectividad, eficiencia y correctitud del proceso de negocio soportado por el workflow

- Facilitar el análisis y definición del workflow antes de poner éste en producción.

Basándose en estas consideraciones surge la necesidad de ayudar a los diseñadores en la tarea de definir workflows y adaptarlos en forma acorde, considerando cambios en tiempos de diseño. Para construir dicha adaptación sobre los workflows, basada en contextos, es necesario definir un metamodelo para expresar restricciones sobre el workflow y segundo, definir otro metamodelo para especificar la adaptación del workflow mediante el soporte de composición. Estos metamodelos deberían capturar el workflow en términos de controles y flujos de datos.

A nivel de workflows se presentan propuestas de adaptación de workflows que son soluciones a bajo nivel, desaprovechando los enfoques que ofrece la Ingeniería de Software para abordar este problema desde un nivel de abstracción mayor como un problema de modelado: en [50][49] se introduce el concepto de *utility function* para expresar una propiedad del workflow que se busca maximizar (rendimiento por ejemplo), de tal forma que la adaptación de un workflow se expresa como un problema de optimización de un atributo en particular. En [78] se propone una red de Petri (*Dataflow-Constrained Workflow Net*) para verificar el comportamiento del workflow ante los cambios de los flujos de datos. En [40] se plantea la adaptación de workflows mediante el descubrimiento de patrones frecuentes de ejecución en los workflows, donde la anticipación de una configuración final de un workflow se soporta en patrones comunes en ejecuciones previos del workflow, usando algoritmos de minería de datos para realizar dichos descubrimientos.

SOA contempla el ensamblaje de servicios atómicos para crear procesos complejos de negocio; dicho ensamble es acompañado por orquestaciones de servicios (establecidas por arquitectos SOA) como mecanismos de composición de alto nivel. La esencia de la orquestación puede usarse para soportar la evolución de orquestaciones de servicios a través de un proceso de combinación de comportamiento en donde un modelo expresa tanto orquestaciones como evoluciones [62].

La aplicación de aspectos a la orquestaciones de servicios (que representan workflows de procesos de negocio) ha sido abordada por investigadores que ven en este paradigma una oportunidad de capturar composiciones de servicios en forma modular, de tal forma que la composición en si misma se hace más dispuesta a adoptar cambios dinámicos. Las técnicas basadas en

aspectos direccionan explícitamente la modularización de intereses cruzados o transversales. En [21] los autores proponen el uso de aspectos como un mecanismo complementario al proceso de composición de servicios a nivel de BPEL, permitiendo una mayor modularidad y adaptabilidad en workflows construidos bajo BPEL. En [26] se expone cómo una aplicación manejada por procesos de negocio, que usa un motor basado en BPEL con capacidades de adición de aspectos, puede ser dinámicamente adaptada con nuevos comportamientos incluso en tiempo de ejecución, soportando requisitos y procesos no contemplados inicialmente por la aplicación.

Las aproximaciones basadas en aspectos permiten abordar atributos de calidad (requisitos no funcionales) desde tempranas etapas del proceso de desarrollo de software, de tal forma que dichos atributos son mapeados en forma de patrones independientes a cualquier aplicación o dominio; dichos patrones se convierten en intereses cruzados (*crosscutting concerns* en terminología AOD) cuya representación podría implicar problemas de confusión. En [70] se presenta una propuesta basada en técnicas MDD para reducir el esfuerzo de aplicar sistemáticamente soluciones reusables que satisfacen requisitos no funcionales, combinando principios orientados a aspectos para mejorar la modularización de los intereses cruzados propios de los patrones que representan dichos requisitos; el trabajo plantea un mapeo desde requisitos aspectuales hacia arquitecturas de software orientadas a aspectos, soportando dicha transición mediante reglas de transformación propias de MDD. Este trabajo utiliza el método **ADORE** (ver sección 3) como framework base para modelar comportamientos aspectuales de workflows y derivados desde atributos de calidad (seguridad). La intención es extender/complementar **ADORE** de tal forma que sea posible soportar a nivel de modelado las adaptaciones de procesos de negocio generadas por la incorporación de comportamientos derivados desde operaciones concretas de seguridad.

Particularmente para el atributo de calidad de la seguridad se evidencian trabajos en donde se promueve la identificación de este tipo de requisitos desde etapas tempranas del proceso de desarrollo; Okubo y Tanaka en [66] proponen una técnica para identificar aspectos de seguridad en un análisis de requisitos usando una extensión de UML conocida como *misuse cases*, la cual representa comportamientos no esperados y actores que poseen restricciones de seguridad (por ejemplo usuarios autorizados - no autorizados). En [63] los autores proponen un perfil (profile) UML para especificar aspectos de seguridad, soportado en un metamodelo que define la incorporación de

seguridad en un diseño basado en UML. Finalmente, en [77] se propone un lenguaje orientado a aspectos para separar los requisitos funcionales y los atributos de calidad en procesos de negocio, identificando los aspectos como las propiedades no funcionales que son transversales a múltiples servicios; dicho lenguaje es soportado por un framework MDD; tanto el framework como el lenguaje son definidos por los autores como metamodelos en *Eclipse Modeling Framework*.

1.3. Objetivos y preguntas de investigación

El objetivo general de esta propuesta es *establecer mediante el uso de técnicas de transformación de modelos y el enfoque de aspectos, la incorporación, en tiempo de diseño, de mecanismos de adaptación sobre workflows con el propósito de considerar restricciones de seguridad sobre los datos y las estructuras de control que hacen parte de un workflow.* Los objetivos específicos del trabajo se relacionan a continuación:

- Definir la integración entre estándares de seguridad basada en control de acceso y métodos de orquestación aspectual de servicios basados en ADORE.
- Seleccionar una semántica básica para representar el mapeo de perspectivas estructurales propias de estándares de seguridad, con perspectivas funcionales propias de los mecanismos para modelado aspectual de orquestaciones de servicios.
- Especificar a nivel de diseño de workflows, el tratamiento de la composición entre modelos aspectuales que representan operaciones de seguridad definidas por estándares, y modelos de comportamientos (orquestaciones de servicios)
- Validar, en un caso de estudio, la integración de modelos aspectuales formulados a partir de estándares de seguridad, con modelos aspectuales que representan comportamientos de un proceso de negocio.

Las preguntas de investigación a resolver en este trabajo son las siguientes:

- ¿Cuál es la forma de aplicar y/o enriquecer el método **ADORE** para considerar la seguridad basada en estándares?

- ¿Cuál es el aporte de los lenguajes de modelado basado en UML para representar, a alto nivel de abstracción, la seguridad como un aspecto y las intervenciones de éste sobre la funcionalidad base?
- ¿Cuál es la importancia de los mecanismos de visualización para analizar la complejidad de las adaptaciones de seguridad propuestas?

Capítulo 2

Fundamentos conceptuales de la adaptación de workflows

A continuación se relacionan los referentes conceptuales utilizados en este trabajo:

2.1. Separación de intereses

El concepto de *separación de intereses* o **concerns** por su equivalente en inglés, definido éste como un subconjunto lógicamente coherente de la funcionalidad global del sistema [36], ha sido planteado desde tempranas épocas del surgimiento formal de la industria del desarrollo de software, destacándose definiciones como en [31] donde establece la separación como la *habilidad para identificar, encapsular y manipular sólo las partes del software que son relevantes a un concepto, meta o propósito en particular*[1]. La construcción de software debe considerar el manejo de contextos de negocio complejos bajo el principio de garantizar la eficiencia y rendimiento de las organizaciones en términos de flexibilidad, reutilización, desacoplamiento, integración y agilidad. La separación de intereses utilizando técnica basadas en aspectos ha sido ampliamente propuesta y aceptada como un mecanismo para encapsular características funcionales-no funcionales en módulos reutilizables a lo largo de una solución, permitiendo incrementar el nivel de abstracción mediante la identificación de características comunes. En [70] se propone el concepto de *Early Aspect* para resaltar la identificación y separación de intereses comunes en los niveles de especificación de requisitos y definición de la arquitectura

[1] http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html

del sistema mediante la combinación de principios del desarrollo orientado a aspectos y desarrollo manejado por modelos o *MDD*. Este tipo de iniciativas promueven la existencia de procesos de negocio cuyas funcionalidades permean a los demás procesos de la organización sin ser repetidos e invasivos, perteneciendo así a un ecosistema de procesos colaborativos con contratos establecidos a través de protocolos de cooperación definidos en los aspectos y las composiciones generadas a partir de éstos. La separación de intereses permite la identificación, estructuración y definición de reglas de negocios a partir de modelos y elementos de alto nivel, con lo cual es posible desarrollar servicios como reglas de negocio a partir del modelado de éstas, la implementación de las reglas como aspectos, y la composición de servicios usando principios orientación a aspectos [21] .

El uso de aspectos es principalmente aplicado al tratamiento de atributos de calidad, de tal forma que los procesos de negocio gestionados en un workflow consideren características adicionales a la funcionalidad. Para promover la reusabilidad de servicios como intereses comunes es importante separar las propiedades funcionales- no funcionales de diferentes aplicaciones, debido a que éstas usarán los servicios en diferentes condiciones de atributos de calidad (por ejemplo, diferentes políticas de seguridad). En la mayoría de prácticas SOA contemporáneas orientadas a la separación de intereses, las propiedades relacionadas con atributos de calidad son especificadas por cada servicio; esta estrategia implica que los desarrolladores y arquitectos SOA deben asegurarse de configurar adecuadamente los atributos de calidad en un conjunto de servicios (generalmente cada atributo de calidad cubre múltiples servicios simultáneamente). Un ejemplo sería la aplicación de una propiedad de seguridad sobre los servicios que participan en un proceso de negocio de compras. La especificación consistente y la validación de propiedades no funcionales a lo largo de un conjunto de servicios propios de un proceso de negocio de considerable complejidad es un proceso tedioso, costoso y propenso al error. Por lo tanto, se hace necesario especificar atributos de calidad para procesos de negocio en lugar de servicios directamente, reduciendo así los costos y sobrecargas en el desarrollo y mantenimiento de aplicaciones.

Desafortunadamente *BPMN* y *BPEL* no soportan formalmente esta práctica de separación de intereses por procesos de negocio[77] . El uso de aspectos gestionados por modelos permite flexibilizar las orquestaciones de servicios desde un alto nivel de abstracción, de tal forma que los procesos de negocio son enriquecidos con características adicionales [26] sin representar mayores

modificaciones o cambios a nivel de código fuente.

El paradigma SOA fomentó la comunicación y colaboración entre servicios y procesos de negocio donde principalmente se mapean funcionalidades específicas. Luego emergen paradigmas de composición y orquestación de servicios principalmente soportados por estándares como *BPEL* y workflows de negocio. En [52] se identifican problemáticas frecuentes propios de entornos de producción de software en términos de procesos de negocio transversales, procesos de negocio comunes y enmarañamiento de reglas de negocio, resaltándose la independencia entre unidades de negocio donde, a falta de una visión sistémica, se automatiza el caos gracias a la falta de cohesión entre procesos, la repetición de procesos, la interpretación diferente de políticas de acuerdo a los contextos donde éstas son aplicadas, y presencia de entropía en el código fuente encargado de gestionar reglas de negocios (que con frecuencia se suele replicar en otros módulos donde sea necesario reutilizar dicha regla).

Este trabajo propone la aplicación del principio de separación de intereses, conjuntamente con el principio de orientación a aspectos presentado en la sección 2.2, con el fin de diferenciar explícitamente los intereses funcionales de los workflows, de los intereses o concerns derivados específicamente desde el atributo de calidad de la seguridad basada en control de acceso.

2.2. Aspect Oriented Software Development (AOSD)

El paradigma de *desarrollo de software orientado a aspectos - AOSD* surge como una evolución de las técnicas tradicionales de Ingeniería de Software, orientándose hacia la separación avanzada de intereses o *concerns*. La orientación a aspectos implica la modularización de los *concerns*, la invocación implícita de éstos como unidades básicas (aspectos), y la definición de mecanismos de entrelazado e invocación de los *concerns*; todo esto aplicado a lo largo del ciclo de vida del desarrollo de software, incluyendo etapas tempranas donde se hace levantamiento de requisitos y análisis-diseño.

En [11] se presenta el resumen de los términos básicos manejados en el *AOSD*, como son:

- *Concern*: area de interés o énfasis en un sistema, importante para un stakeholder del sistema.

- *Crosscutting*: término usado para indicar que los *concerns* pueden ser dispersos y enlazados unos con otros sobre una representación particular. La *dispersión* implica ue un *concern* puede extenderse por varios módulos. El *enlazado* implica que un módulo puede contener contiene partes de varios *concerns*.
- *join point*: es un elemento definido en un lenguaje sobre el cual es posible adjuntar o vincular un comportamiento adicional.
- *pointcut*:define un conjunto de *join points*.
- *advice*: describe el comportamiento adicional a ser ejecutado sobre un *joint point* en particular, definiendo así una parte del comportamiento transversal (*crosscutting*).

Los *Advices* y los *pointcut* son usados conjuntamente: el *pointcut* especifica en qué *joint point* tiene que ser ejecutado el comportamiento de un *advice*.

Johan Brichau y Theo D'Hondt plantean una interesante analogía para representar el concepto de la *separación de intereses* y administración de los mismos mediante aspectos[2], utilizando como símil las diferentes características de una casa (ver Figura 2.1). En dicha gráfica se expone la necesidad de manejar cada *concern* en forma separada, con el fin de considerar:*i)* todas las posibles descomposiciones que se pueden presentar en un sistema (evitando descomposiciones basadas generalmente en una dimensión como la funcionali-dad), *ii)* todos los posibles paradigmas de construcción que intervienen en la gestión de los *concerns*, y *iii)* la aplicación de la separación aspectual desde etapas de análisis y diseño evitando el abordaje de los aspectos sólo en etapas de implementación.

Actualmente se evidencia la aplicabilidad del paradigma orientado a aspectos en técnicas de modelado cuyo propósito es identificar, diseñar y establecer las relaciones de intereses transversales (*crosscutting concerns*) desde etapas tempranas del proceso de desarrollo. Una de las técnicas de modelado aspectual más popular (empleada en este trabajo) es el lenguaje *Theme/UML* [3], el cual está enmarcado dentro de la propuesta denominada *Model-Driven Theme/UML*[19]. Dicha propuesta define un proceso de desarrollo dividido en tres fases: la fase de *modelling* representa la especificación de los intereses bases, intereses o concerns aspectuales y las relaciones entre ellos usando el lenguaje *Theme/UML*; la fase de *Composition* se encarga

[2]http://www.info.ucl.ac.be/ jbrichau/courses/introductionToAOSDPresentation.pdf
[3]http://www.dsg.cs.tcd.ie/aspects/themeUML

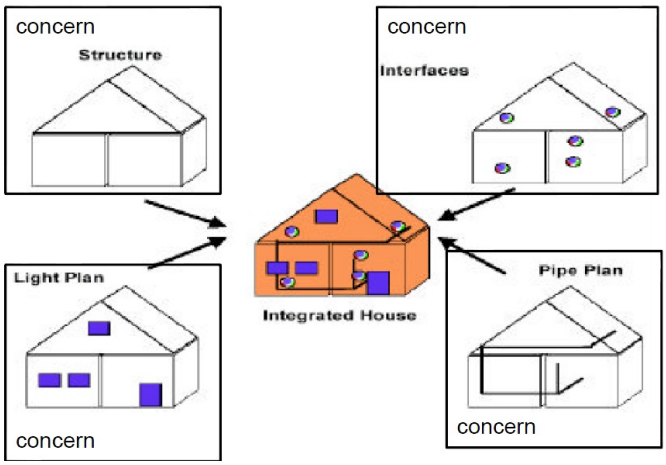

Figura 2.1: Metáfora para *separación de concerns* propuesta por Brichau y D'Hondt

de componer los elementos modelados usando un plug-in Eclipse que genera una vista en UML del sistema modelado; la fase de *Transformation* se encarga de la generación de código a partir del modelo obtenido en la fases previas. La figura 2.2 presenta un resumen general del proceso *Model-Driven Theme/UML*[4], incluyendo una etapa inicial de *modelo de dominio* y *especificación de requisitos*.

2.3. Workflows y adaptaciones

El manejo de la complejidad del software y de sistemas de información, acorde con el principio de Dijkstra, implica la identificación de un conjunto de funcionalidades o características relacionadas y el posterior empaquetamiento de las mismas en subsistemas o entidades lógicas con responsabilidades e interfaces claramente identificadas. La *separación de intereses* facilita la reutilización a nivel de *granularidad rústica* dado que la especificación de unidades (cualquiera sea el nombre con el que éstas sean identificadas - servicios, subsistemas, módulos, etc.), sean utilizadas por diferentes aplicaciones. La *separación de intereses* también facilita la respuesta a cambios, convirtiéndose así en un importante mecanismo para soportar flexibilidad en el

[4]Fuente: https://www.cs.tcd.ie/Eamonn.Linehan/lero/mdthemeuml_big.png

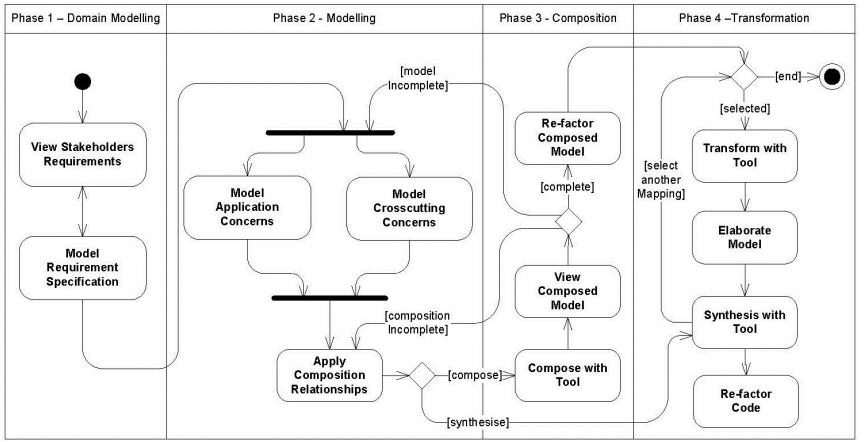

Figura 2.2: Resumen general *Model-Driven Theme/UML*

software o procesos de negocio (las entidades pueden ser modificadas o intercambiadas con otras entidades, pero se continúa suministrando la misma funcionalidad dado que las interfaces permanecen estables).

La gestión de procesos de negocios *BPM* implica la definición y el modelado de procesos de negocio core de una organización, alieando los aspectos organizacionales con las necesidades de los clientes con el fin de promover la eficiencia y efectividad de la misma La fundamentación conceptual de la definición y modelado de procesos de negocio se encuentra en el principio de *separación de intereses* o *concerns* identificado por Dijkstra (*centrar la atención sobre aspectos*).

La creación y administración de procesos de negocio es soportada por **workflows**[5], definidos éstos como la automatización de procesos de negocio, total o parcialmente, durante la cual los documentos, la información o las tareas son comunicados entre los participantes, de acuerdo a un conjunto de reglas procedimentales[6]. Los workflows son gestionados por *workflow management systems*, los cuales son software encargados de definir, crear y administrar la ejecución de workflows, ejecutando uno o varios motores de

[5]http://unpan1.un.org/intradoc/groups/public/documents/un-dpadm/unpan040198.pdf , slide 23

[6]Workflow Management Coalition Terminology & Glossary, Document Number WFMC-TC-1011.

workflows encargados de interpretar la definición del proceso de negocio, interactuar con participantes del workflow, y realizar invocaciones que utilizan aplicaciones o herramientas tecnológicas para gestionar atributos de calidad (por ejemplo, un motor de reglas de seguridad). La tecnología de los workflows soporta procesos de negocio enmarcados dentro de un sistema software dado o entre un conjunto de sistemas software, integrando efectivamente dichos sistemas.

La principal consideración expuesta por los workflows de procesos de negocio consiste en que éstos son construidos a partir de la orquestación de múltiples actividades/servicios. Dichos servicios normalmente se ejecutan sobre diferentes sistemas de cómputo usando diversas combinaciones de invocaciones síncronas/asíncronas. Adicionalmente, estas actividades o servicios podrían a su vez ser encadenadas a otras actividades/servicios, dando lugar a una secuencia de procesamiento bajo un orden específico. Por lo tanto, para las empresas es crítico administrar la coordinación de las actividades en el orden requerido para cumplir con las metas del negocio [48].

El diseño e implementación de workflows puede ser apreciado como un tipo de programación de alto nivel, donde las funcionalidades y la gestión de atributos de calidad caracterizan los bloques que son organizados dentro de un workflow. La importancia lograda por los workflows es la *representación explícita de estructuras de procesos de negocio en modelos de procesos*, de tal forma que es posible ajustar controladamente los procesos de negocio acorde con dichos modelos. Las técnicas de modelados de procesos de negocio son usadas para proveer representaciones explícitas de las relaciones entre aplicaciones empresariales y entre secciones de workflows. Los modelos de procesos de negocio proveen las bases conceptuales para definir cuándo y bajo qué condiciones las aplicaciones empresariales son invocadas en el contexto de escenarios de integración [80].

Los cambios en ambientes y contextos de negocio han creado la necesidad de generar estrategias de adaptación para soportar la administración eficiente y efectiva de los procesos de negocio. Como resultado, existen numerosos intentos para mejorar sistemas de información a través de funciones avanzadas de administración de procesos negocio, que van más allá de la simple manipulación de tareas independientes [4].

El problema de la adaptación de los workflows ha sido ampliamente abor-

dado tanto por los workflows de negocio como por los workflows científicos[7], destacándose la elaboración de iniciativas particulares que proponen técnicas, frameworks o modelos conceptuales para adaptar workflows. La mayoría de estas propuestas se basan en el abordaje de aspectos funcionales del workflow, dejando a un lado el tratamiento de los requisitos no funcionales aplicados a la adaptación del workflow para centrarse en la orquestación de servicios disponibles en ambientes distribuidos y heterogéneos, siendo éste el principio más cercano a lo que se conoce como sistemas *context-aware* o *context-services*[71]. Algunas de las técnicas de adaptación de workflows presentadas por Smanchat [72] tienen como principal característica la adaptación de workflows basada en cambios en el contexto derivados de necesidades del usuario en cuanto a la incorporación de nuevos servicios o el reemplazo de servicios existentes. Los trabajos relacionados en la revisión de Smanchat sugieren la adopción del principio de *punto de variación*, propio de la ingeniería de dominio de las líneas de producción, aplicados a servicios específicos de un workflow que son posibles candidatos de adaptación.

Principios tomados del *Aspect Oriented Software Development - AOSD* y *Model Driven Development - MDD* facilitan un alto grado de flexibilidad: AOSD puede ser aplicado para identificar intereses comunes o *concerns*, visualizando los escenarios donde éstos pueden ser aplicados a lo largo del proceso de negocio que se automatiza en un workflow; los modelos de procesos de negocio pueden ser adaptados para cumplir nuevos requisitos; además, las modificaciones a los modelos de procesos puede ser aplicadas en forma inmediata para ajustar los procesos de negocio.

2.4. Adaptaciones de workflows basadas en seguridad

Los workflows de procesos de negocio son construidos mediante la combinación de servicios a través del uso de un lenguaje de especificación de procesos, el cual permite determinar las tareas a ser ejecutadas y el orden de ejecución de dichas tareas. '*WS-BPEL* (*Web services Business Process Execution Language*)[8] se ha convertido en el lenguaje estándar para implementar procesos de negocio basado en servicios. Sin embargo, a pesar de los

[7] se refieren a workflows que involucran el procesamiento de datos aplicados en sectores como la medicina, biología, bioinformática y ciencias de la vida en general.

[8] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

progresos significativos que han conducido al desarrollo de un lenguaje para procesos de negocio, los cambios importantes relacionados con la gestión de atributos de calidad necesitan ser direccionados antes de que los sistemas *BPM* sean usados en entornos altamente distribuidos y en servicios, para responder a situaciones no contempladas inicialmente en el proceso de negocio, por ejemplo, *cómo puede influir en la ejecución de un proceso la presencia de participantes adicionales?*. Para efectos de este trabajo, las operaciones de *seguridad basadas en control de acceso* serían los participantes adicionales cuya incorporación al workflow es implementada como una modificación a éste.

WS-BPEL no provee soporte alguno para la especificación de políticas o restricciones de autorización sobre la ejecución de actividades de un proceso de negocio. Algunos autores proponen extensiones *WS-BPEL* para soportar este tipo de consideraciones, como *BPEL4People*[9], desafortunadamente dichas propuestas abarcan un segmento reducido del problema a considerar, dejando parte de la problemática en función del abordaje que se pueda aplicar con *WS-BPEL*. Debido a la amplía adopción de los web services para implementar procesos de negocio complejos y de *WS-BPEL* como lenguaje estándar para especificar procesos de negocio basados en servicios, el problema de *cómo asociar usuarios autorizados con actividades de un proceso de negocio* surge como una interesante pregunta de investigación a la cual actualmente se siguen formulando soluciones de diferente tipo, que en su gran mayoría no consideran como tal la adaptación del workflow de proceso de negocio, ya que estas soluciones se formulan desde el punto de vista de *WS-BPEL* (inclusive teórico) más no del workflow en sí mismo. Por ejemplo, los autores en [79] proponen un modelo de control de acceso denominado *BPEL4RBAC*, conjuntamente con una extensión para *WS-BPEL* llamada *BPEL4RBAC policy language*, con el fin de incorporar seguridad basada en roles en *WS-BPEL*. Otra propuesta de incorporación de control de acceso basada en extensiones de *WS-BPEL* se encuentra en el *modelo de autorización RBAC-WS-BPEL* propuesto en [10] para soportar diferentes tipos de restricciones basadas en*RBAC* como perfiles *WS-BPEL*.

Algunas propuestas de incorporación de contro de acceso directamente en los workflows son (en orden cronológico):

[9]Puede consultarse
http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/
y http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bpel4people

- El *workflow authorization model (WTA)* propuesto en [3] que soporta la especificación de autorizaciones de cómo los sujetos obtienen acceso a objetos requeridos durante la ejecución de una tarea, sincronizando un flujo de autorizaciones con el workflow como tal. La solución *WAT* es implementada usando redes de Petri.

- La ejecución secuencial de tareas propuesta en [9] donde se especifica una dupla tarea-rol para identificar una tarea del workflow y una lista de roles autorizados para ejecutar dicha tarea, especificando también el número máximo de activaciones de esa tarea en una instancia del workflow.

- El método de especificación de restricciones de autorización propuesto en [27], donde se define la separación de restricciones relacionadas con obligaciones, la vinculación de restricciones, obligaciones relacionadas con la antigüedad relativa de los usuarios que realizan diferentes tareas, y restricciones determinadas por información contextual basada en el usuario.

Diferentes modelos y mecanismos para abordar la gestión de autenticación, control de acceso y privacidad han surgido y evolucionado con el tiempo, impulsados éstos por una gran variedad de factores, desde avances tecnológicos tales como los dispositivos biométricos, hasta las necesidades crecientes de seguridad y privacidad. Los *sistemas operativos* proporcionan mecanismos básicos para la identificación, autenticación y control de acceso. Los *sistemas manejadores de base de datos* también ofrecen soporte para almacenar políticas de control de acceso. Sin embargo, los mecanismos ofrecidos tanto por sistemas operativos como sistemas de bases de datos no son extensibles, por lo que cada vez que un entorno de aplicación requiere controles de autenticación o de acceso más sofisticados las aplicaciones deben incluir la lógica explícita para dichos controles. Por lo tanto, este enfoque conduce a la duplicación de esfuerzos y mayores costos en el desarrollo y mantenimiento del software.

Debido a estas razones, han surgido modelos y mecanismos para controlar el acceso de manera independiente de las aplicaciones y procesos de negocio, tales como *XACML* (ver sección 4.1). Al mismo tiempo, la necesidad de direccionar el comportamiento de un sistema de control de acceso o un sistema de autenticación a través de políticas ha fomentado el desarrollo de diferentes modelos y mecanismos de gestión de políticas. El uso de un enfoque basado

en políticas aumenta la flexibilidad, reduce los costos de desarrollo de software, y simplifica la gestión de la seguridad. Los cambios en los requisitos de seguridad, simplemente implican modificar las políticas sin requerir cambios en el software, el control de acceso y en los mecanismos de autenticación. Un enfoque importante para el abordaje de la seguridad está representado por el desarrollo de los servicios de seguridad basados en políticas, proporcionando con ello las funciones de gestión de seguridad relevantes para los procesos de negocio. Este enfoque particularmente prometedor cuando los procesos se organizan de acuerdo con el paradigma SOA. Una cuestión importante a tener en cuenta cuando se diseñan técnicas de seguridad basadas en SOA es la *forma de lograr la coordinación entre los múltiples servicios de seguridad.* La seguridad es una tarea compleja que no puede ser alcanzado por un único servicio, se necesita coordinar múltiples servicios entre sí.

A continuación se relacionan dos ejemplos de adaptación de workflows acordes con seguridad: en [28] los autores desarrollaron una infraestructura flexible para control de acceso en aplicaciones colaborativas soportadas por Grid computing, mediante la separación explícita de la evaluación de políticas y la administración del workflow, combinando éstos posteriormente en puntos de decisión de forma tal que la ejecución del workflow es usada para rastrear cambios en el contexto de seguridad que son dependientes del dominio de la aplicación, políticas definididas de ejecución, o atributos de usuarios y/o servicios. La administración de seguridad es realizada utilizando el *GAAA Toolkit* y para la administración del workflow se utiliza *Globus*. En [69] se propone un framework para realizar control de acceso basado en workflows (*WBAC*), donde el workflow como tal se usa para capturar las tareas en entornos clínicos relacionadas con diagnósticos, terapias, tratamientos, y administración de hospitales. Dado el nivel de seguridad que se debe consi-derar en el tratamiento de la información de pacientes y médicos, los autores optaron por separar el módulo de control de acceso en un repositorio de políticas montando bajo la plataforma *Ponder2*[10] que soporta la especificación de políticas; el workflow de actividades se implementa usando el workflow management system denominado *YAWL*. Ambos trabajos previos exponen la separación explícita de la ejecución del workflow con el tratamiento de las políticas de control de acceso, integrando ambas partes posteriormente mediante invocación de servicios.

Este trabajo utiliza los principios relacionados en las secciones 2.3 y 2.4

[10] http://ponder2.net/

con el fin de proponer adaptaciones de workflows acorde con operaciones de control de acceso (sección 4), desde altos niveles de abstracción del workflow (nivel de modelo).

2.5. Model Driven Development (MDD)

Acorde con el planteamiento de Cabot[11], *MDD* es un paradigma de desarrollo de software que usa modelos como los *artefactos primarios* del proceso de desarrollo. Normalmente en *MDD* la implementación es generada semiautomáticamente desde dichos modelos. Esta definición se deriva de [54], en donde se propone la noción de que se construye el modelo de un sistema y posteriormente este modelo puede transformarse en cosas reales (código por ejemplo).

Zhu y Liu en [83] plantean que *MDD* introduce implícitamente la *separación de intereses o concerns* a través de diferentes tipos/capas de modelos; esto en virtud del uso de los modelos como artefactos de ingeniería primarios a lo largo del ciclo de vida de un desarrollo de software, y los niveles de abstracción alcanzados por el uso de modelos. En consecuencia, el uso de modelos permite aplicar análisis a nivel de *early aspects* (sección 2.3), logrando adaptibilidad, portabilidad y reusabilidad gracias a los modelos. Las actividades del desarrollo de software, incluyendo evaluaciones, son más sistemáticas y formales gracias a la aplicación de *MDD*.

En [35] se relacionan cinco características claves que los modelos deberían poseer:

- *Abstracción*: un modelo siempre es una renderización reducida del sistema que éste representa.
- *Comprensibilidad*: Un modelo no es suficiente solamente para abstraer detalles, tambien debe presentar consideraciones restantes en una forma específica (por ejemplo una notación) que apele directamente a la intuición.
- *Precisión*: un modelo debe proveer una representación fideligna de las características de intereses propias del sistema bajo modelado.

[11] http://modeling-languages.com/blog/content/relationship-between-mdamdd-and-mde

- *Predictivo*: los modelos pueden ser usados para predecir correctamente propiedades interesantes no obvias (evidentes) del sistema modelado, sea a través de experimentación (por ejemplo, mediante la ejecución de un modelo sobre un computador) o a través de algún tipo de análisis formal.
- *Economía*: un modelo debe ser significativamente más económico para construir y analizar que el sistema modelado.

Los objetivos del *MDD* son: *i)* incrementar el nivel de abstracción para el desarrollo de software, *ii)* reducir el tiempo de desarrollo, *iii)* mejorar la calidad del software reduciendo el tiempo de pruebas, *iv)* reducir el costo de mantenimiento y el costo total de propiedad de aplicaciones empresariales[12].

La aplicación de MDD en este trabajo se evidencia mediante la propuesta de adaptación de workflows que es direccionada por modelos gracias al uso del método relacionado en el capítulo 3. De esta forma es posible considerar adaptaciones de workflows desde altos niveles de abstracción en fases tempranas de construcción del workflow.

[12]http://www.utdallas.edu/ chung/RE/Intro-to-MDD-UTD.pdf

Capítulo 3

El Método ADORE

El paradigma SOA - *Arquitectura Orientada a Servicios* - soporta el ensamblaje de servicios atómicos para crear aplicaciones que implementan procesos de negocio complejos. El ensamble de servicios puede llevarse a cabo mediante orquestaciones de servicio definidos por los arquitectos SOA [60][62]. Según SOA, una aplicación es *un conjunto de servicios que implementa un proceso de negocio de misión crítica.*

La complejidad implícita por abordar los intereses o *concerns* no ortogonales[1] durante el diseño de aplicaciones SOA puede ser controlada efectivamente utilizando técnicas de modelado que soporten la separación y composición de intereses. El uso de estas técnicas permite que los arquitectos y desarrolladores modelen orquestaciones que entrelacen y coordinen comportamientos no ortogonales a través de un conjunto de servicios.

Desafortunadamente, las herramientas y formalismos tradicionales para modelado de procesos (*BPMN, BPEL*) no proveen mecanismos de separación de intereses necesarios para componer aspectos no ortogonales de procesos de negocio, dado que ambos son planteamientos tecnológicamente orientados que utilizan un enfoque de diseño en gran escala; dicho enfoque genera una inversión de esfuerzo significativa para desarrollar y adaptar aplicaciones ro-

[1]Nota: Para definir el término non-orthogonal concern o intereses no ortogonales se utilizará la definición dada por [1], donde un *concern* o interés de comportamiento se define en términos de separación, distinción y autonomía semántica. La separación significa que cada interés o concern es disjunto de otros intereses. La distinción significa que cada interés tiene características únicas, donde éste tiene sola una misión. Un concern es semánticamente autónomo cuando éste puede ser comprendido sin la existencia de otros concerns.

bustas que involucran muchos servicios orquestados en diversa variedad de formas [61].

3.1. ADORE

El método **ADORE** (*Activity moDel suppOrting oRchestration Evolution-modelo de actividad para soportar evolución de orquestaciones*[2]) [58][61][62], definido por el grupo de investigación **MODALIS** de la Universidad de Nice-Sophie Antipolis[3], provee una técnica composicional para modelar orquestaciones de servicios complejas. Los modelos describen pequeñas orquestaciones las cuales son posteriormente entrelazadas para producir un modelo que representa una orquestación de un amplio conjunto de servicios [62]. Los modelos de pequeñas orquestaciones, llamados *fragmentos* de orquestación, describen diferentes aspectos de un proceso complejo de negocio, donde cada aspecto responde a un interés en particular que puede ser un comportamiento funcional o un comportamiento direccionado por un atributo de calidad. Los fragmentos son orquestaciones no concebidas para ejecución directa; el objetivo de los fragmentos es ser integrados en las orquestaciones. En la terminología **ADORE** un fragmento es considerado como un *proceso incompleto*.

El método permite a los arquitectos SOA modelar orquestaciones complejas de servicios mediante la composición de modelos de orquestaciones más pequeñas (fragmentos). El método **ADORE** también se utiliza para entrelazar fragmentos que administran nuevos intereses o concerns dentro de modelos existentes de una aplicación. Los aspectos encargados de manejar intereses o concerns no funcionales podrían ser tratados como fragmentos genéricos o reutilizables (de forma análoga al concepto de non-orthogonal concern), por lo tanto, se podrían requerir orquestaciones de servicios donde se involucra el entrelazado de aspectos en diferentes puntos, en uno o varios servicios.

ADORE puede ser visto como un método que integra *MDD* y *Modelado Orientado a Aspectos (AOM)* para apoyar el desarrollo de aplicaciones *SOA*. Desde la perspectiva del *Aspect Oriented Modeling*, los fragmentos son aspectos y **ADORE** usa mecanismos aspectuales (*join points, pointcuts*, y *advises*) para determinar qué, dónde y cómo componer respectivamente sobre

[2] http://www.adore-design.org/doku/
[3] http://modalis.i3s.unice.fr/

dichos fragmentos. **ADORE** permite a los expertos en procesos de negocio modelar diferentes aspectos del proceso por separado y luego componer éstos.

La Figura 3.1 presenta los metamodelos fundamentales del método **ADORE** especificados en [58]; estos metamodelos definen el concepto de *BusinessProcess* como un conjunto de variables y actividades con sus respectivas relaciones de ordenamiento. El metamodelo define también otros elementos importantes en el contexto de **ADORE**, como el concepto *Orchestration* (realización de un comportamiento asociado a una operación o servicio), el concepto *Fragment* (comportamiento incompleto que será integrado en otro proceso), y el concepto *Activity* (uso de variables de entrada y asignación de resultados a variables de salida) con sus respectivos tipos (invocación, asignación, reporte de falla, recepción de mensajes, envío de respuesta y actividades de sincronización - Nop). El concepto *Universe* define los procesos de negocio disponibles para los algoritmos de composición.

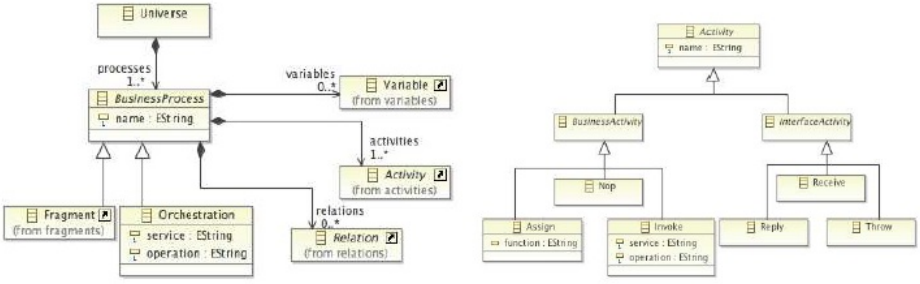

Figura 3.1: Metamodelos **ADORE**. Fuente:[58].

El soporte para la separación de intereses ayuda a dominar la complejidad implícita al crear y evolucionar modelos de grandes procesos de negocio en donde deben tratarse múltiples comportamientos funcionales y no funcionales. **ADORE** puede ser visto como un método que integra *MDD* y *Modelado Orientado a Aspectos (AOM)* para apoyar el desarrollo de aplicaciones SOA. **ADORE** utiliza directivas de composición para ligar los fragmentos a conjuntos de actividades. Un enfoque orientado a aspectos también establece por lo menos dos mecanismos de composición: *(i)* entrelazado de aspectos (para integrar aspectos dentro de un código base), y *(ii)* el ordenamiento de aspectos (para definir y ordenar aspectos entrelazados alrededor del mismo *joint point*). **ADORE** introduce una variación con respecto al segundo mecanismo de composición mencionado, definiendo un algoritmo para

componer fragmentos alrededor de un *join point* compartido en lugar de ordenar dichos fragmentos. El proceso de transformación de modelos usando el método **ADORE** se demuestra mediante *i)* el ingreso por parte de los expertos del proceso de negocio, de modelos textuales que contienen información sobre las invocaciones de servicios, comportamientos incompletos y composición de orquestaciones, y *ii)* la generación automática de modelos de orquestaciones en donde se evidencia tanto los comportamientos incompletos como sus entrelazados con orquestaciones de servicios.

3.2. Ejemplo de fragmentos y orquestaciones ADORE

En [61] los autores definen un conjunto de orquestaciones y fragmentos **ADORE** por medio de los cuales se componen los servicios que soportan el *CCCms* (caso de estudio definido en [46] y relacionado en la sección 5.5), usando un enfoque SOA. Los autores definen orquestaciones para representar casos de uso del *CCCms*, y fragmentos **ADORE** para especificar extensiones o flujos alternos de casos de uso. De igual forma, algunos atributos de calidad son realizados como comportamientos del sistema definidos por medio de fragmentos; un ejemplo se presenta en la Figura 3.2, en donde se define un fragmento especial para un requisito no funcional de seguridad definido en [46], el cual se resume en la re-autenticación de un usuario *si éste no se encuentra activo en un periodo de 30 minutos.*

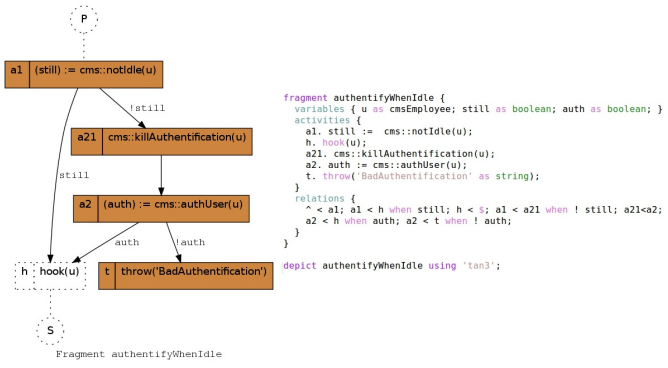

Figura 3.2: Fragmento **ADORE** para requisito de seguridad (reautenticación) del *CCCms*

La Figura 5.9 presenta una orquestación **ADORE** definida para el caso de uso *Execute Rescue Mission* especificado en [46]; dicha orquestación relaciona los pasos ejecutados durante el transporte de una víctima de un accidente de tránsito, al hospital más apropiado.

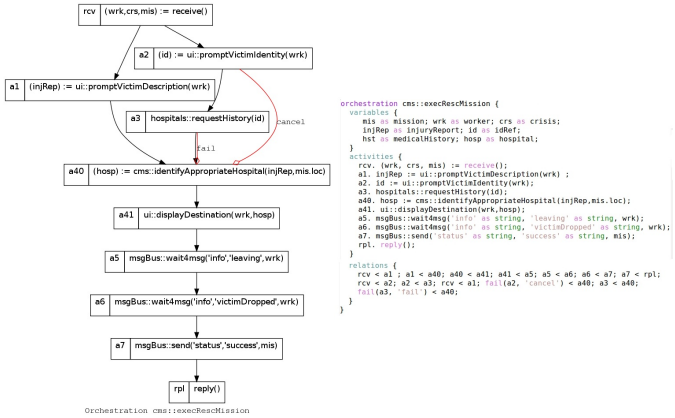

Figura 3.3: Orquestación **ADORE** para el caso de uso *Execute Rescue Mission* del *CCCms*

De acuerdo al método **ADORE**, un **fragmento** es una *orquestación de servicios armable*, es decir, una orquestación que puede ser conectada en otras orquestaciones. Un fragmento corresponde a un interés específico y usa un punto de vista parcial correspondiente a su objetivo concreto. **ADORE** define tres actividades especiales obligatorias para cualquier fragmento: una actividad especial denominada **hook** representa dónde el fragmento será conectado en una orquestación existente. Una actividad **P** re-presenta los predecesores del *hook*, y una actividad *S* representa los sucesores del *hook*. Un fragmento puede usar a priori entidades externas desconocidas, dichas entidades son modeladas como parámetros de fragmentos. Durante la composición de fragmentos y orquestaciones, estos parámetros son resueltos mediante su reemplazo sintáctico con referencias a entidades definidas. Esta parametrización permite a los diseñadores definir fragmentos genéricos que pueden ser instanciados y usados en diferentes contextos [61].

Del documento de especificación del *CCCms* formulado por [46] y desarrollado bajo el método **ADORE** en [61] se destaca principalmente la propuesta de abordar los intereses alusivos a requisitos no funcionales co-

mo fragmentos de tal forma que éstos sean conectados en orquestaciones de servicios más grandes. Este tipo de abordaje, en donde se encuentra una considerable complejidad conceptual en el diseño de sistemas, es la principal motivación para usar técnicas para modelado de orquestaciones de servicios como **ADORE**, teniendo en cuenta que el comportamiento de la complejidad varía, ya que ésta se reduce cuando se concentra el esfuerzo en un fragmento o un grupo de fragmentos, pero al mismo tiempo se incrementa al observar globalmente el modelo de orquestaciones de servicios como un todo.

La separación de intereses ha sido ampliamente aceptada como un mecanismo para encapsular características funcionales/no-funcionales dentro de módulos reutilizables sobre una solución software. Esta técnica incrementa el nivel de abstracción mediante la identificación de características comunes (lo cual es un objetivo de la implementación de procesos de negocio.

Sin embargo, el diseño de fragmentos de atributos de calidad expuesto por [61] presenta invocaciones genéricas establecidas a partir de las especificación de los requisitos no funcionales. Como se ha mencionado anteriormente, los atributos de calidad abordados en el diseño de un workflow podrían contener estándares y modelos previamente definidos y que idealmente deberían ser consideradas en el diseño de la encapsulación aspectual del atributo usando fragmentos **ADORE**. Por lo tanto, debe considerarse la especificación del contexto del atributo de calidad en términos de estándares, invocación de servicios y definición conceptual, para su posterior diseño e incorporación dentro de un workflow de procesos de negocio. La consideración de las especificaciones conceptuales de los comportamientos de atributos de calidad permitiría enriquecer la formulación conceptual de los fragmentos **ADORE** de tal forma que éstos podrían administrar comportamientos especializados derivados de atributos de calidad, a nivel de modelo, que posteriormente son entrelazados en orquestaciones de servicios que componen procesos de negocio complejos.

Este trabajo propone que un artefacto *RBAC-XACML* puede ser considerado como un interés o *concern*, y su aplicación podría ser extendida a un conjunto de servicios, aplicaciones o procesos de negocio, si existe un mecanismo para adicionar composiciones de servicios sobre un proceso de negocio en particular, y considerar la seguridad a nivel de políticas de control de acceso basado en roles.

Capítulo 4

Seguridad basada en control de acceso

El *control de acceso* es una estrategia para protección de información de un sistema realizada mediante la definición de *roles* asociados a *permisos* de acceso sobre *recursos* específicos. La autorización de acceso se describe explícitamente a través de *políticas* o *permisos*. El concepto de *rol* agrupa a los usuarios que cumplen con un perfil concreto para efectos de ejecutar operaciones sobre el sistema. Generalmente el *control de acceso* incluye subcaracterísticas de seguridad establecidas en la norma ISO 25010 (Software product Quality Requirements and Evaluation -SQuaRE), como la *autenticación* (grado en el que se comprueba la identidad de un sujeto o recurso), *conformidad* (adherencia a estándares, convenciones o regulaciones, como el modelo *RBAC* y/o el estándar *XACML* presentados en la sección 4.1), y la *integridad* (grado en el que un sistema o componente previene accesos no autorizados a, o modificaciones de, programas o datos).

La gestión de la funcionalidad y los atributos de calidad (particularmente seguridad) podría ser realizada usando un enfoque aspectual de composición aplicado en niveles superiores de abstracción, en donde sea posible definir composiciones basadas en aspectos bajo un mismo principio o manejador arquitectónico sin necesidad de recurrir a diferentes tecnologías que soportan un mismo estándar, o a diferentes modificaciones - extensiones de estándares previamente concebidos.

4.1. RBAC y XACML

El modelo **RBAC** - *Role Based Access Control*[1] es una aproximación gene-ralizada al control de acceso donde los roles representan funciones definidas en una organización y los permisos (definidos como *la habilidad o derecho de ejecutar alguna acción sobre un recurso bajo ciertas condiciones previamente especificadas*) dependen de los roles en lugar de los usuarios. Las autorizaciones asociadas a un rol están estrictamente relacionadas a los objetos de datos y los recursos necesarios para ejecutar las funciones asociadas al rol. Los usuarios son simplemente autorizados para jugar al rol apropiado. El principal beneficio de adoptar *RBAC* radica en la simplificación de las ta-reas de definición de políticas de seguridad para expertos del negocio que no tienen conocimiento de específico de las plataformas software para soportar operaciones de seguridad. Los esfuerzos de administración del sistema son minimizados debido al reducido número de relaciones requeridas para relacionar roles con permisos.

El *lenguaje extensible de etiquetas para el control de acceso - eXtensible Access Control Markup Language* o **XACML**[2] - es un estándar de OASIS para especificar y forzar políticas de control de acceso independientes de alguna plataforma software. XACML (actualmente en versión 3.0) se ha convertido en el estándar de facto para especificar políticas de control de acceso sobre web services. Desde el año 2004 OASIS definió un profile *XACML* para *RBAC* con el fin de vincular soluciones prácticas RBAC en ambientes de web services [3]. La administración de privilegios en un entorno distribuido usando soluciones basadas en *RBAC* proporciona un control mayor de éstos, disminuyendo también la complejidad implícita en este proceso de administración. El profile *XACML-RBAC* hace posible el manejo de políticas de seguridad en un entorno distribuido gracias a la localización de roles y políticas desde sitios distribuidos que pueden ser administrados en forma independiente [33].

XACML proporciona un conjunto estándar de elementos XML para la formulación de políticas de control de acceso, un protocolo solicitud/respuesta para consultas relacionadas, y un modelo de flujo de datos abstractos entre los componentes funcionales. En su núcleo, la arquitectura general de

[1] http://csrc.nist.gov/groups/SNS/rbac/

[2] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[3] Puede consultarse el sitio http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cd-03-en.pdf para encontrar la definición del profile *XACML-RBAC* acorde con la versión actual de *XACML* definida por OASIS

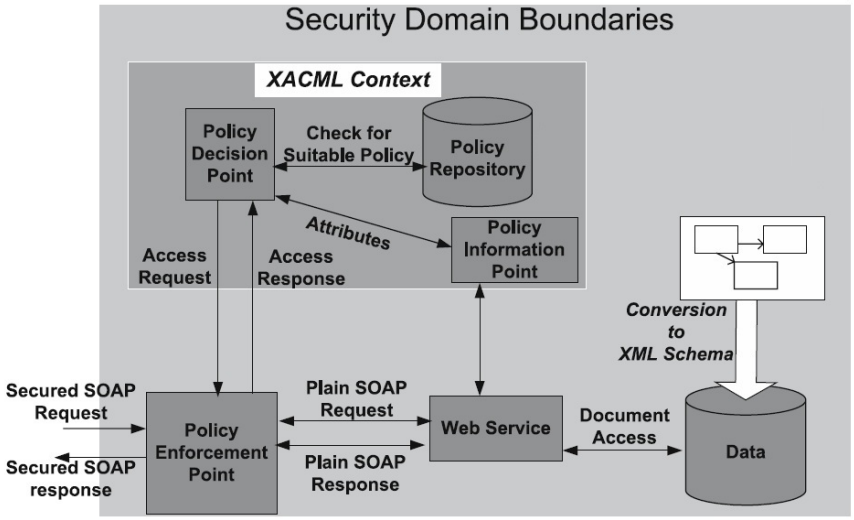

Figura 4.1: Arquitectura general de un proceso de control de acceso XACML. Fuente: [15]

XACML (presentada en la Figura 4.1) define un conjunto de servicios básicos encargados de implementar el control del acceso a recursos. Estos servicios están relacionados a una infraestructura específica de seguridad, cuyos componentes implementan requisitos de seguridad como autenticación, autorización, no repudiación, confidencialidad e integridad [15]. La arquitectura *XACML* define la presencia de tres componentes fundamentales en la resolución de accesos a recursos: el *Policy Enforcement Point (PEP)*, el *Policy Decision Point (PDP)* y el *Policy Information Point (PIP).*

XACML define requisitos de control de acceso para recursos protegidos de una aplicación. El lenguaje de decisión de accesos *XACML* se utiliza para representar una consulta que pregunta *si una determinada acción se debe permitir sobre un recurso.* Si se encuentra una política asociada al recurso consultado, los atributos en la solicitud se comparan con los atributos que figuran en las reglas de política. Por último se establece una respuesta donde se determina si la solicitud se debe permitir o no, utilizando para ello uno de cuatro valores: Permitido (*Permit*), Denegado (*Deny*), Indeterminado (*Indeterminate* - se produjo un error o se perdió algún valor requerido, por lo que no se puede hacer una decisión) o No aplica (*Not Applicable* - la solicitud no puede ser respondida por un servicio).

Un proceso de control de acceso soportado por *XACML* involucra los siguientes pasos:

1. Un usuario envía un elemento *PEP* una solicitud de acceso sobre un recurso .

2. El *PEP* forma una solicitud con los atributos del sujeto, del recurso y de la acción, y envía dicha solicitud a un *PDP*.

3. El *PDP* analiza la solicitud y obtiene una política o un conjunto de políticas que apliquen a la solicitud desde un almacén (repositorio) de políticas.

4. El *PDP* consulta al servicio *PIP* para obtener los valores de los atributos relacionados con el sujeto, el recurso, o el contexto, que se encuentren en la política, y recibe una respuesta con valores de dichos atributos.

5. El *PDP* compara los atributos de la solicitud contra los atributos contenidos en las reglas de la política y formula una respuesta acerca de si debería permitirse o no el acceso. Esa respuesta se envía al *PEP*, el cual puede permitir o denegar el acceso al solicitante [44]

La figura 4.2 presenta el modelo de especificación para *XACML 2.0*[4] publicado por OASIS. Como tal OASIS no utliza el término *metamodelo* para referirse a dicho modelo, sin embargo en este trabajo se considerará como metamodelo dado que éste cumple con las condiciones MOF para especificación de modelos a alto nivel de abstracción.

4.2. Ejemplo de implementación de control de acceso bajo JBoss

La figura 4.3 presenta un resumen de una implementación típica para aplicar políticas de control de acceso usando una plataforma de desarrollo basada en herramientas *JBoss* que soportan la definición de procesos de negocio bajo principios *BPM-BPEL*. El esquema de la solución propuesta considera tres componentes:

[4]Disponible en http://docs.oasis-open.org/xacml/2.0/XACML-2.0-OS-ALL.zip

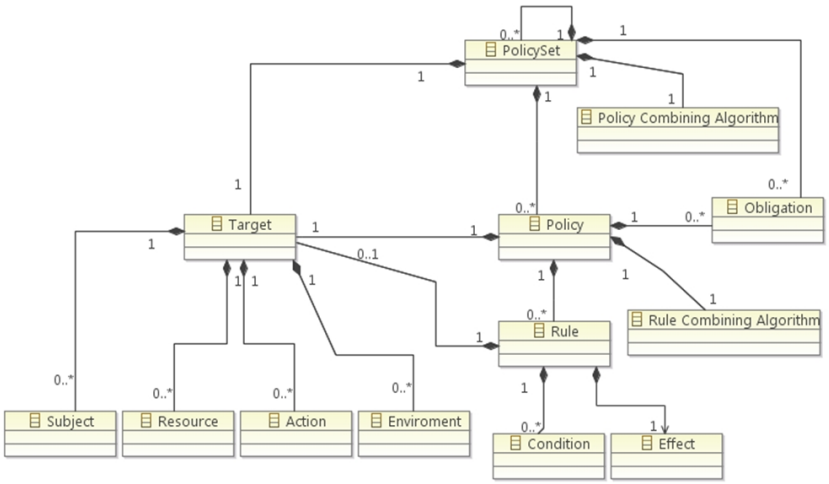

Figura 4.2: Modelo de especificación para XACML definido por OASIS

- Un framework de seguridad para gestionar las políticas de control de acceso usando el proyecto *JBoss PicketBox*[5] , que posee un motor compatible con Oasis *XACML* v2.0.
- Un motor de reglas de negocio basado en el proyecto *Drools* de JBoss[6], con el fin de administrar las reglas de negocio que condicionan la ejecución de los procesos de negocio propios de un workflow.
- Un módulo basado en *JBoss Riftsaw*[7], el cual provee un motor compatible con *WS-BPEL 2.0* para definir procesos de negocio que orquestan web services.

El uso de frameworks y proyectos propios de una plataforma tecnológica para desarrollo de aplicaciones empresariales permitiría deducir la obtención de cierto grado de uniformidad para gestionar tanto los procesos de negocio como las políticas de control de acceso. Sin embargo esta aproximación evidencia cómo se involucran tres frameworks distintos, propios de un mismo referente software como *JBoss*, para afrontar la administración dinámica de

[5]http://www.jboss.org/picketbox
[6]http://www.jboss.org/drools/drools-expert.html
[7]http://jboss.org/riftsaw

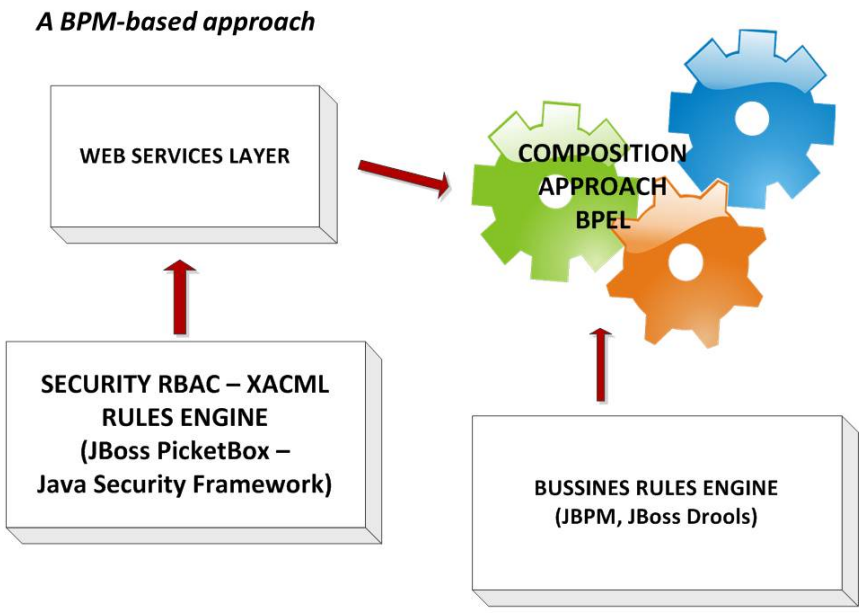

Figura 4.3: Ejemplo de ambiente para gestión de control de acceso bajo JBoss

los procesos de negocio (bajo reglas de negocio), la administración de políticas de control de acceso y la orquestación de los web services. De incluirse la aplicación de aspectos para soportar la *separación de intereses* las alternativas serían *JBoss AOP*[8] o *Aspectj*[9], dada la tecnología particular considerada. No obstante, se evidencia que el manejo de múltiples frameworks para abordar este tipo de soluciones adiciona problemas de complejidad, dependencia y mantenibilidad en el código fuente que soporta los procesos de negocio.

El esquema de implementación propuesto responde a problemas comunes en aplicaciones empresariales, donde generalmente las soluciones a condiciones cambiantes de negocio implican la intervención en el código fuente directamente, reflejándose allí mezclas de reglas de negocio con reglas de seguridad relacionadas con control de acceso.

[8] http://jboss.org/jbossaop
[9] http://www.eclipse.org/aspectj/

El uso de frameworks especializados permite a los servicios administrar modularmente consideraciones formuladas a partir de la ejecución del proceso de negocio, usando orquestaciones y composiciones que facilitan la adaptación del proceso sin afectar el código fuente de la solución, por ejemplo, orquestaciones de servicios donde sea posible adicionar autenticación basada en diferentes técnicas (usuario/clave, autenticación por huella, autenticación por dispositivos biométricos, etc), autorizaciones por roles que tienen accesos a determinadas opciones del sistema y a recursos del mismo, opciones de seguridad basadas en jerarquías de roles (por ejemplo acceso y ocultamiento parcial-total de información) manejadas todas éstas como reglas dinámicas de negocio.

La nueva preocupación, desde el punto de vista de la implementación, se refleja en la cantidad de frameworks de implementación requeridos para abordar los requisitos formulados sobre los procesos de negocio. Implícitamente se deduce la necesidad de contar con un mecanismo a alto nivel de abstracción, el cual permita administrar la incorporación de diversas tecnologías durante las orquestaciones de servicios encargadas de definir los comportamientos asociados a los workflows de procesos de negocio.

4.3. Integración del control de acceso, AOSD y MDD

Un enfoque para resolver el problema planteado por la adaptación de los workflows acorde con restricciones de seguridad, al menos a nivel de diseño, es el uso de modelos soportadas por principios aspectuales como el *entrelazado* o *weaving*, de tal forma que, partiendo de un modelo de procesos de negocio sea posible incorporar reglas de seguridad. Idealmente estos modelos deberían estar acordes con uno o varios metamodelos, como metamodelos de procesos de negocio, metamodelos de seguridad, y metamodelos de integración de propuestas. De esta forma sería posible incorporar elementos de seguridad en los modelos de procesos de negocio para soportar adaptaciones de éstos sobre un alto nivel de abstracción.

En [82] los autores definen un mapping entre los metamodelos de *BPMN* y *XACML* para permitir la especificación de políticas de control de acceso basadas en *XML*. Una de las características más sobresalientes consideradas en el mapeo definido (presentado en la Figura 4.4), es el uso del elemento *BPMN authorization constraint* del metamodelo de *BPMN*, que contiene información sobre las actividades a ser ejecutadas por un *punto de aplicación de*

políticas dependiendo de la salida de un *proceso asociado para evaluación de políticas*. Este mapeo en particular (*BPMN authorization constraint - Obligation XACML*) implica que el metamodelo *BPMN* define, desde su más alto nivel de abstracción, un elemento conceptual para considerar implicaciones de seguridad basada en *control de acceso*.

En [42] los autores presentan un modelo conceptual basado en el metamodelo de UML 2.0, donde se define conjuntos y relaciones usadas para forzar control de acceso en SOA, combinando estándares basados en *Role Based Access Control (RBAC)* y *Attribute Based Access Control (ABAC)*. En [55] los autores presentan una técnica para integración de *BPEL* y *RBAC* a nivel de metamodelo. Esta propuesta se deriva de dos principios básicos, *i)* la falta de direccionamiento del *control de acceso* por parte de *BPEL* (a pesar de que el control de acceso es un aspecto importante e integral de los procesos de negocio), y *ii)* la eficiencia que trae la integración de la ingeniería de roles con el modelado de procesos de negocio. La integración de metamodelos se soporta en el *proceso de ingeniería de roles manejado por escenarios*, donde un escenario contiene una acción y una secuencia de eventos; por lo tanto, un proceso *BPEL* puede ser visto como un conjunto de descripciones formalizadas en escenarios, de tal forma que un rol es un contenedor de permisos sobre operaciones asociadas a un escenario en particular.

Las propuestas mencionadas basadas en integración de modelos permiten deducir:

- El abordaje y mapping de intereses de seguridad desde modelos de procesos de negocio de alto nivel de abstracción.
- La posibilidad de incorporar reglas de seguridad a partir de definiciones del negocio y requisitos de seguridad asociados a los procesos de negocio.
- La independencia del mapping conceptual entre negocio-seguridad con respecto a la tecnología específica a aplicar en una solución concreta (posibilidad de extender dicho mapping sobre diferentes plataformas tecnológicas).

La consideración de características de seguridad a partir de metamodelos refuerza el concepto de *Early Aspect* (sección 2.1), ya que dicho concepto propone la detección temprana y administración de aspectos asociados a requisitos no funcionales, como la seguridad, usando modelos conceptuales de

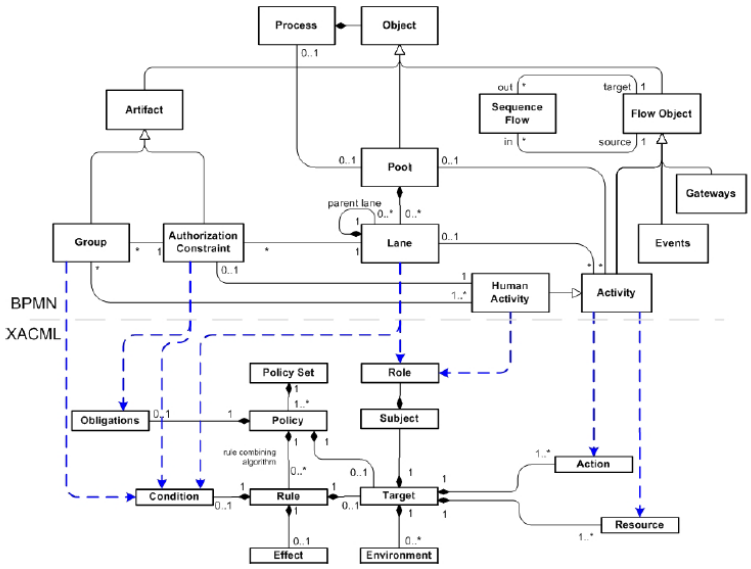

Figura 4.4: Mapeo BPMN-XACML. Fuente [82].

nivel superior basados en *MOF*. La seguridad es quizás la aplicación más exitosa de los conceptos orientados a aspectos. Particularmente al trabajar con el modelo *RBAC* los aspectos capturan condiciones de acceso en forma modular, tales como las condiciones de control propias de políticas *RBAC*. Dichas políticas describen las restricciones que un usuario dado (enmarcado en cierto rol) debe cumplir con el propósito de ejecutar una acción sobre un recurso protegido (acceso en un sistema controlado). En [12] el autor define una transformación de políticas de control de acceso *RBAC* definidas bajo el metamodelo *SecureUML*, a aspectos modelados en un lenguaje específico, es decir, un lenguaje que contiene un subconjunto de instrucciones comunmente encontradas en lenguajes orientados a aspectos.

Un interés o *concern* de seguridad podría ser considerado como un aspecto, de tal forma que la aplicación del mismo podría extenderse a un conjunto de aplicaciones o procesos de negocio, si se cuenta con un mecanismo que permita adicionar (*entrelazar*) aspectos en composiciones de servicios que consideren operaciones de seguridad a nivel de roles y políticas de control

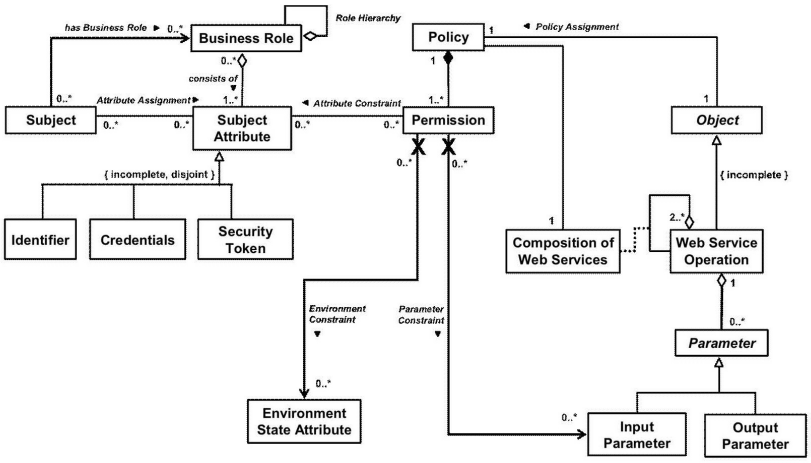

Figura 4.5: Modelo integración RBAC-ABAC en SOA. Fuente [42].

de acceso. Mientras *MDD* y *AOSD* son enfoques diferentes en múltiples formas, *MDD* provee abstracciones específicas, y *AOSD* ofrece mecanismos para modularización y composición de intereses o *concerns*. Estas características permiten obtener beneficios derivados del uso de ambos enfoques usados en forma complementaria [53]. Stein y Hanenberg en [73] exponen que, aunque los enfoques *AOSD* y *MDD* están relacionados con la adaptación de un sistema de entrada con el fin de producir un sistema de salida aumentado/-modificado (usando mecanismos denominados *entrelazado* en *AOSD* y *transformación* en *MDD* respectivamente), ambos enfoques difieren desde que el enfoque orientado a aspectos proporciona abstracciones adicionales que permiten especificar condiciones en tiempo de ejecución dentro de selecciones de *joint points*. Transformaciones sencillas, como realizar una refactorización de un método para cambiar el nombre del mismo, no se podrían lograr a través de técnicas orientadas a aspectos, pero si con técnicas dirigido por modelos.

Un interesante trabajo donde se propone una combinación de *AOSD* y *MDD* se presenta en [23]; allí se aborda el problema de conectar reglas de negocio ejecutables de alto nivel con aplicaciones existentes orientadas a objetos. El autor expone un contexto en donde *i)* las reglas de negocio se entrelazan y dispersan a lo largo de la implementación de aplicación, por tanto, las reglas de negocio se hacen difíciles de localizar, añadir, cambiar o eliminar. *ii)* La reglas de negocio no son comprensibles para los expertos

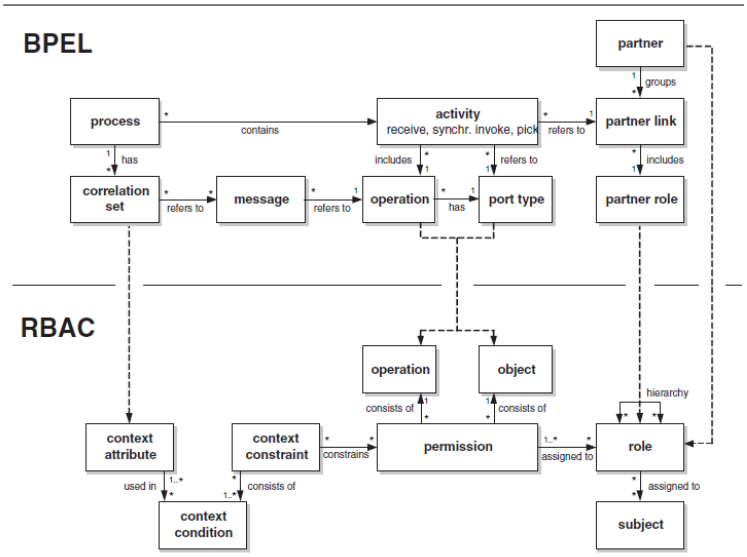

Figura 4.6: Integración metamodelos BPEL-RBAC. Fuente [55].

de dominio quienes a su vez no son partidarios de convertirse en expertos en programación; por esta razón se genera una estrecha conexión entre las reglas de negocio y la implementación existente de la aplicación. El primer problema se direcciona mediante la encapsulación de la regla de conexión en un módulo separado, desacoplando la funcionalidad de las reglas. Este desacoplamiento no es sencillo porque las reglas de conexiones son transversales a la funcionalidad de la aplicación. La programación orientada a aspectos (*AOP*), establece mecanismos de modularización (aspectos) para encapsular el código transversal al tiempo que garantiza la inversión de dependencias entre la aplicación principal y estos aspectos. El segundo problema se aborda mediante la construcción de una capa de abstracción, un modelo de dominio que facilita la expresión de las reglas de negocio en términos de conceptos de dominio. La solución presentada incorpora principios de *MDE* con el fin de lograr la generación automática de implementaciones ejecutables para las reglas de negocio de alto nivel y las conexiones de dichas reglas.

4.4. SecureUML y Model Driven Security

Debido al interés por abordar la seguridad desde tempranas etapas del desarrollo de software se presentan en la comunidad académica numerosos trabajos previos donde se proponen profiles o extensiones UML para representar explícitamente requisitos de seguridad. Jurjens [45] propone un enfoque para el desarrollo de sistemas seguros mediante una extensión de UML llamada *UMLsec*, permitiendo construir modelos UML que contienen especificaciones formales de requisitos de seguridad, como *confidencialidad*, o el flujo seguro de información. En [24] se propone un perfil *RBAC* con el cual se puede modelar gráficamente especificaciones de control de acceso conjuntamente con la especificación del dominio del problema desde el principio de la fase de diseño. La propuesta de [24] particularmente utiliza *OCL* para validar la formación y significado de los modelos versus *RBAC*.

Entre las alternativas más representativas para modelado de la seguridad basadas en UML sobresale el proyecto *Model Driven Security*[7], una metodología para el desarrollo de sistemas seguros en el dominio de control de acceso, que incluye entre sus principales premisas *i)* el uso de metamodelos basados en *MOF* para definir los principales elementos de un sistema seguro acorde con *RBAC*, y dialectos para formalizar y combinar lenguajes de modelado, *ii)* un lenguaje de modelado (con su respectivo metamodelo MOF base) para soportar la generación de políticas de control de acceso, y *iii)* técnicas para generar infraestructuras de control de acceso. Basin et al en [6] [7] definen un metamodelo denominado *SecureUML*, como extensión del metamodelo UML, donde incorporan términos propios de *RBAC* como tipos propios en el metamodelo (tipos como *Usuario*, *Rol*, *Permiso* y sus respectivas relaciones). Los autores muestran como UML es usado para especificar información asociada con control de acceso en el diseño global de un software, y cómo esta información puede ser usada para generar automáticamente infraestructuras completas para control de acceso; de esta forma se combina sintáctica y semánticamente UML con propuestas de modelado de seguridad *RBAC* para formalizar requisitos de control de acceso.

La fundamentación conceptual de *Model Driven Security* radica en la semántica que proveen los modelos conjuntamente con las políticas de control de acceso, proporcionando una base para la generación de implementaciones acordes con los metamodelos y para el análisis de las políticas a nivel de modelado. Particularmente esta iniciativa plantea un metamodelo cen-

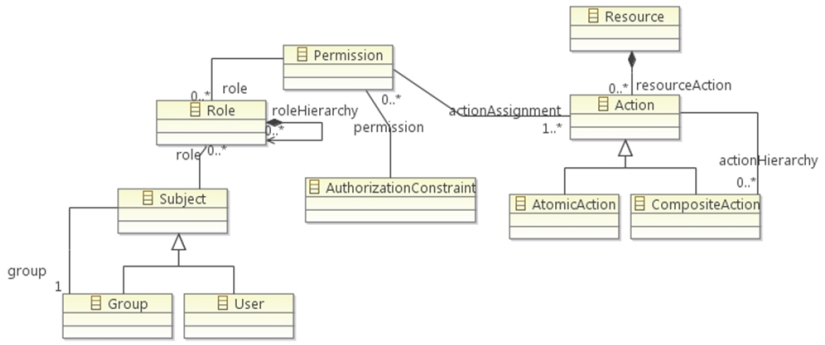

Figura 4.7: Metamodelo SecureUML. Fuente [7]

tral para *SecureUML*, donde se formaliza las decisiones de control de acceso dependientes de dos tipos de información: *i)* decisiones de control de acceso declarativas que dependen de *información estática* como la asignación de usuarios y permisos a los roles (configuración *RBAC*),y *ii)* decisiones de control de acceso programáticas que dependen de *configuraciones dinámicas*, como la satisfacción de restricciones de autorización en el estado actual del sistema. Cuando los modelos poseen una semántica formal es posible hacer consultas sobre las propiedades de dichos modelos (particularmente en este caso, propiedades *RBAC*), entendiendo las consecuencias sutiles de las políticas definidas en éstos[6].

En [14] se presenta una realización muy interesante del *Model Driven Security* sobre workflows inter-organizacionales en el contexto de web services y orquestación de servicios. La descripción de los requisitos de seguridad se realiza en un alto nivel de abstracción. Los artefactos de seguridad relevantes se generan para una arquitectura específica. La realización soporta la transición sistemática de los requisitos de seguridad, a través de la generación de artefactos de seguridad, hacia una solución segura basada en una plataforma de web services. La especificación de los requisitos de seguridad se realiza de manera independiente de la plataforma y por lo tanto puede ser aplicado por expertos del dominio sin un conocimiento técnico profundo. Los autores en [25] reportan un caso de aplicación de *SecureUML* en un proyecto real de industria, donde obtuvieron las siguientes lecciones aprendidas:

- La construcción de modelos durante la etapa de análisis de requisitos y diseño del sistema provee un fundamento para análisis temprano y

detección de fallas.

- Los modelos del diseño de la seguridad integran modelos de seguridad con modelos de diseño del sistema, permaneciendo reusables, evolutivos e independientes de plataformas tecnológicas.
- Los modelos de diseño de seguridad en *SecureUML* son comprensibles para quienes están familiarizados con la notación basada en UML.

La identificación temprana de intereses cruzados o *concerns* derivados desde atributos de calidad, hace posible identificar, a nivel de modelo, los puntos de adaptación de los workflows de procesos de negocio. Los metamodelos propuestos para abordar términos propios de los diversos tipos de operaciones de seguridad enriquecen la especificación conceptual tanto del proceso de negocio como de los intereses transversales que expresan como aspectos a ser incorporados en composiciones de servicios.

Capítulo 5

Integración del estándar *XACML* en ADORE

5.1. Administración de control de acceso bajo ADORE

ADORE permite especificar fragmentos en donde se invoquen servicios que aborden aspectos funcionales de un proceso de negocio, o manejen comportamientos derivados de atributos de calidad. El soporte composicional de fragmentos permite que éstos sean combinados acorde a las demandas del proceso de negocio, de tal forma que los fragmentos en sí mismos podrías ser reutilizados por otros procesos de negocio.

La Figura 5.2 ilustra una propuesta de orquestación y composición de fragmentos direccionada por **ADORE**. Dicha propuesta se fundamenta en el uso intensivo de fragmentos para direccionar reglas de negocio (condiciones que implican variaciones en el comportamiento funcional del proceso de negocio), y reglas de control de acceso; los fragmentos son posteriormente compuestos usando el mismo principio composicional de **ADORE**, evitando así recurrir a otras tecnologías para obtener articulación de fragmentos y/o servicios tal como se mencionaba en la sección 4.2.

La propuesta basada en **ADORE** hace posible la redefinición del esquema formulado a partir de principios *BPM-BPEL* con su respectivo soporte tecnológico (expuesto en la Figura 4.3), de tal forma que, aplicando el principio de composición de fragmentos **ADORE**, se direcciona la formulación de fragmentos específicos orientados al manejo de las políticas de control de acceso *RBAC-XACML*, en forma genérica e independiente de la tecnologÃa

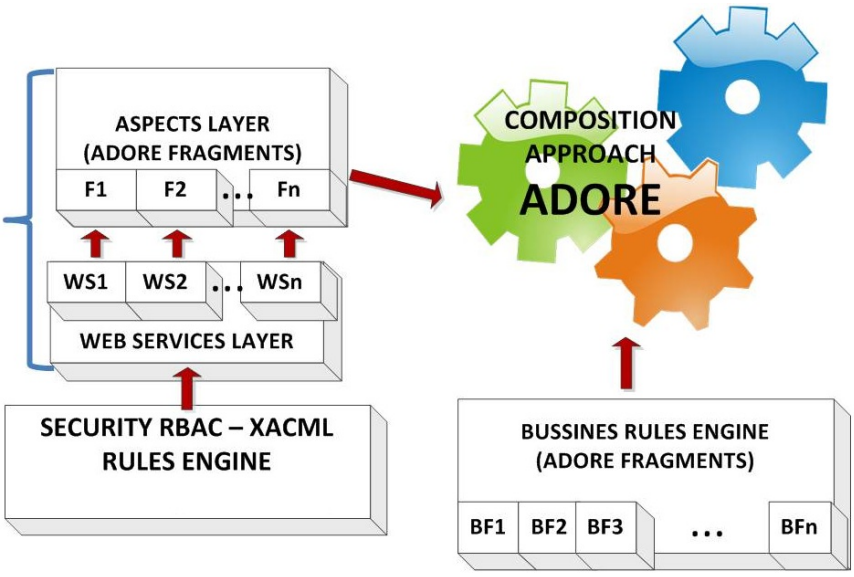

Figura 5.1: Propuesta de orquestación y composición de fragmentos genéricos en **ADORE**

subyacente, (dependiendo sólo de las interfaces de los servicios que implementan las operaciones de seguridad).

Debe tenerse en cuenta que **ADORE** trabaja a niveles de modelo para establecer los términos de composición y orquestación de servicios propios de un proceso de negocio. Por lo tanto, el código fuente de los servicios como tal no es abordado directamente en **ADORE**; en lugar de ello, el uso de frameworks dedicados a seguridad, y en general cualquier código fuente que sea requerido en el proceso de negocio es comprometido directamente por los servicios que **ADORE** se encarga de orquestar y componer.

El esquema propuesto permite deducir varias conclusiones preliminares:

1. Los servicios podrían usar cualquier plataforma tecnológica base para acceder y manejar reglas de control de acceso *RBAC-XACML*, dado que la orquestación de servicios no interviene directamente el código fuente que implementa el control de acceso de los roles por parte de frameworks especializados. Los principios aspectuales involucrados por **ADORE** en las orquestaciones y composiciones se aplican a nivel de

modelos de procesos de negocio.

2. Los frameworks especializados para ejecutar operaciones de control de acceso tienen implícitamente un mapeo con el metamodelo *SecureUML* (sección 4.4). Dicho mapeo podría ser usado en altos niveles de especificación del proceso de negocio para orquestar fragmentos dedicados a manejar políticas de control de acceso, usando modelos UML acordes con *SecureUML* y modelos de procesos de negocio (estrategia de integración de modelos soportada por MDD).

3. El uso de un método de alto nivel como **ADORE** permite reducir la complejidad en la administración de las orquestaciones y composiciones de servicios, evitando la sobrecarga generada por el uso de múltiples frameworks especializados a nivel de implementación.

5.2. Integración de metamodelos ADORE, *SecureUML* y *XACML*

La formulación de fragmentos **ADORE** que incorporan elementos de comportamiento derivados desde atributos de calidad (como el *control de acceso* para el caso específico de la seguridad), no contemplados inicialmente en la especificación de fragmentos **ADORE** (metamodelo MOF definido en [58]), implica una integración a nivel de metamodelos MOF para soportar conceptualmente la definición de este tipo de fragmentos, enriqueciendo el método **ADORE** en sí mismo.

Como se ha mencionado anteriormente, *SecureUML* es una técnica MDE para especificación de requisitos de control de acceso basado en roles sobre sistemas software, utilizando técnicas de metamodelado para soportar RBAC con restricciones de autorización especificadas en *Object Constrain Languaje - OCL*. Diversos autores han sugerido considerar conjuntamente *SecureUML* y *XACML* de tal forma que los modelos reflejen condiciones de control de acceso en *SecureUML* que posteriormente sean mapeadas en *XACML*, logrando incluso cierto nivel de automatización al generar reglas de transformación desde los modelos UML-*SecureUML* a reglas *XACML* [47][82].

En [43] los autores exponen que *SecureUML* no ofrece ningún mecanismo para especificar las estrategias de resolución de conflictos entre políticas, mientras que en *XACML* dicha resolución es clara y concreta. Adicionalmente, *SecureUML* no soporta el agrupamiento de políticas. Este agru-

pamiento mejora el rendimiento de un sistema, dado que es posible excluir ciertos grupos de políticas de los controles de autorización. Tanto la resolución de conflictos y la agrupación de política son consideraciones cruciales en la estructuración y gestión de las políticas en sistemas complejos. En [47] los autores presentan una técnica MDE para desarrollar auto-protección en sistemas autónomos, donde se utiliza modelos *SecureUML* con el fin de especificar el control de acceso y soportar el diseño de reglas de auto-protección para reaccionar a vulnerabilidades de seguridad; posteriormente, aplicando reglas de transformación, se generan políticas *XACML* a partir de los modelos y reglas especificadas bajo *SecureUML*.

A partir del esquema **ADORE** presentado en la figura 5.2 y considerando que el metamodelo de *SecureUML* requiere el uso de un lenguaje adicional para soportar el modelo y validación de controles de acceso (tales como *OCL*), se propone la integración de los metamodelos de **ADORE**, *SecureUML* y *XACML* para establecer desde un alto nivel de abstracción las especificaciones de un fragmento **ADORE** encargado de abordar el manejo de seguridad basada en roles y control de acceso.

La figura 5.2 expone la integración de dichos metamodelos donde se define una *EClass* en *EMF* denominada *Secure XACML Fragment*, que hereda todas las propiedades de un fragmento **ADORE** (a nivel de proceso de negocio y a nivel de actividad), y tendrá *hook, predeccessors y successors* tal como un fragmento **ADORE** común. Adicionalmente, al concepto *Secure XACML Fragment* se le añaden relaciones de composición con elementos definidos en el metamodelo *XACML* para especificar que un fragmento **ADORE** de seguridad basada en control de acceso tendrá: *i)* un sujeto que inicia la consulta de acceso sobre un recurso dado, *ii)* el recurso sobre el cual se está solicitando el acceso, *iii)* la política o conjunto de políticas relacionadas al recurso en cuestión.

Para asegurar la trazabilidad de los elementos *XACML* con los requisitos de seguridad basada en control de acceso especificados para el workflow de procesos de negocio, se establece una relación de mapeo (*maps*) entre los elementos del metamodelo de *XACML* y elementos del metamodelo de *SecureUML*; con esta relación de mapeo se pretende establecer que, cualquier requisito de seguridad basada en roles y control de acceso definido para un workflow, puede ser mapeado en forma inmediata a elementos *XACML* administrados por un fragmento aspectual **ADORE**. Idealmente, las reglas

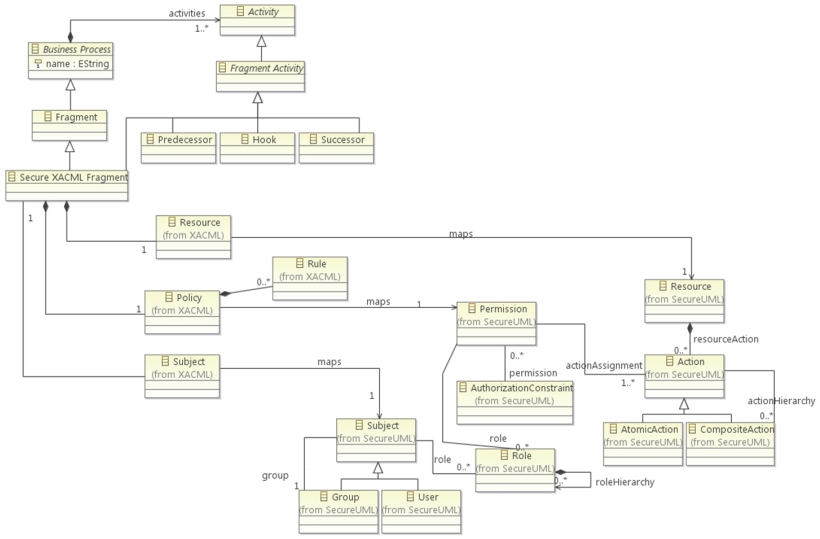

Figura 5.2: Integración metamodelos **ADORE**, SecureUML y XACML

de control de acceso *XACML* deberían ser derivadas de los modelos de seguridad. Por lo tanto, el metamodelo debe reflejar el mapeo de *XACML* a conceptos propios de *SecureUML* con el fin de garantizar que los elementos que intervienen en el control de acceso se derivan desde las fases tempranas de concepción y modelado del sistema.

5.3. Integración de Theme/UML para incorporar semántica en ADORE

El diseño de sistemas como los workflows de procesos de negocio generalmente aborda múltiples vistas para considerar intereses derivados desde diferentes dimensiones (dimensión de funcionalidad, dimensión de calidad, dimensión de negocio, dimensión de datos, dimensión tecnológica, entre otras). Los intereses son compuestos o enlazados para construir una solución. En [58] se resalta cómo el método **ADORE** soporta la composición de entidades manejadas por comportamientos (fragmentos), es decir, las unidades de modelado **ADORE** expresan los intereses de múltiples vistas en términos de comportamientos. Los demos y ejemplos previos implementados en

ADORE modelan comportamientos sin una distinción explícita de la dimensión asociada a cada comportamiento; por lo tanto, es posible que los fragmentos formulados sean limitados a una dimensión funcional acorde con la concepción de *comportamiento* propia del método **ADORE**.

La perspectiva de comportamientos propia del método **ADORE** conduce hacia la definición de entidades funcionales que no consideran vistas independientes derivadas desde comportamientos no funcionales, los cuales definen previamente una estructura conceptual. La definición de fragmentos bajo el método **ADORE** induce a que la información estructural sea inmersa directamente en el comportamiento que se está especificando. **ADORE** no tiene un enfoque de múltiples vistas: los fragmentos son definidos en una solo vista (vista funcional mediante un modelo textual) de tal forma que la estructura está inmersa en dicho modelo. Este trabajo propone el uso de una perspectiva funcional soportada por **ADORE** y una perspectiva estructural soportada por Theme/UML con el propósito de formular fragmentos capaces de abordar comportamientos no funcionales.

Tal como se evidenció en la sección 4.1 con el estádar *XACML*, un atributo de calidad podría contener un conjunto de servicios asociados e infraestructuras o dimensiones específicas, como el caso de tecnologías hardware/software de soporte para operaciones de seguridad. Otros tipos de intereses abordan especificaciones propias de la dimensión, por ejemplo, si un modelo de proceso de negocio debe considerar la administración de reglas de negocio, el fragmento o conjunto de fragmentos formulados deberá considerar operaciones específicas sobre un motor de reglas de negocio.

Adicionalmente, en los fragmentos **ADORE** construidos para los diferentes casos de aplicación del método (relacionados en el sitio oficial del proyecto presentado en la sección 3.1), se evidencia el uso de información propia del contexto modelado en forma de *variables* introducidas directamente en la formulación del fragmento. De acuerdo con el método **ADORE**, el conocimiento del contexto mapeado en los fragmentos está asociado exclusivamente a la descripción del proceso de negocio abordado. Un ejemplo se encuentra en la figura 5.3, donde se presenta un fragmento **ADORE** que contiene información del contexto en el bloque *variables* definido para dicho fragmento. Como se puede apreciar, la información sobre el contexto del fragmento (variables *cmsEmployee, witness, crisisInformation* y *crisisCheckList*) es definida por el experto **ADORE** a partir de la especificación del caso de

uso. Por lo tanto no se observa un mapeo de las características y contexto al fragmento **ADORE** como tal.

```
rchestration cms::captureWitnessReport {
 variables {
   id as crisisIdentifier; coord as cmsEmployee;
   wi as witness; i as preliminaryInformation;
   ccl as crisisCheckList; ci as crisisInformation;
   pi as phoneInformation; exact as boolean;
 }
 activities {
   rcv. (coord, id) := receive();
   a10. wi := ui::promptWitnessInfo(coord);
   a11. cms::setWitness(wi,id);
   a2. i := ui::promptPrelimInfo(coord);
   a2a12. pi := phoneCie::getInfo(wi.id);
   a2a3. exact := cms::validateWitnessInfo(wi,pi);
   a3. ccl := cms::buildCheckList(i);
   a4. ci := ui::promptCheckList(coord,id, ccl);
   a50. cms::assignEmergencyLvl(id,ci);
   a51. msgBus::send('status' as string, 'active' as string,id);
   rpl. reply();
 }
 relations {
   rcv < a10; a10 < a11; a10 < a2; a2 < a3; a3 < a4; a4 < a50;
   a50 < rpl; a10 < a2a12; a2a12 < a2a3; a2a3 < a50; a50 < a51;
 }
}
```

Figura 5.3: Ejemplo de información de contexto en un fragmento **ADORE**.

Con el objetivo de formular un conjunto de fragmentos genéricos para soportar operaciones candidatas a ser incorporadas en múltiples workflows de procesos de negocio, a nivel de modelo, es necesario complementar la definición de fragmentos bajo el método **ADORE** con el uso de técnicas para modelado de múltiples vistas, en aras de combinar comportamientos (**ADORE**) y estructuras que representan contextos de las operaciones. Este trabajo propone el uso del lenguaje de modelado *Theme/UML* (ver sección 2.2), para modelar el contexto de la operación a encapsular en un fragmento **ADORE**. Dicho contexto se compone de la especificación de la operación según un estándar o referente empleado (como *XACML*), y un subconjunto de características del workflow bajo consideración.

Con el propósito de soportar la definición de fragmentos **ADORE** a partir de atributos de calidad cuyos comportamientos son genéricos respecto a los comportamientos propios de un workflow de procesos de negocio, y basados en trabajos como [2][5][19], se ha decidido usar un subconjunto de

los diagramas de la fase de *modelling* establecida en el proceso de desarrollo *Model-Driven Theme/UML* para mostrar el mapeo de la información del *contexto* del atributo de calidad hacia el fragmento **ADORE**. El *contexto* se define en términos de las características o *features* especificadas para el sistema, y el conjunto de estándares que gobiernan la aplicación e invocación de servicios subyacentes, como el estándar *XACML* en este caso.

La figura 5.4 presenta una propuesta para sustentar la especificación de fragmentos **ADORE** genéricos que posteriormente serán aplicados en diferentes workflows de procesos de negocio. El principal objetivo es integrar la técnica de gestión de comportamientos del método **ADORE** con una técnica de *modelado de múltiples vistas* empleada para soportar la información estructural que define los intereses de atributos de calidad (contexto del fragmento). Los intereses o concerns posteriormente serán enlazados con fragmentos de procesos de negocio usando los mecanismos de composición de **ADORE**. El comportamiento transversal (*crosscutting*) modelado en *Theme/UML* se enlaza con otros aspectos o fragmentos mediante la actividad **hook()** definida nativamente en **ADORE** y expuesta en el metamodelo de la figura figura 5.4.

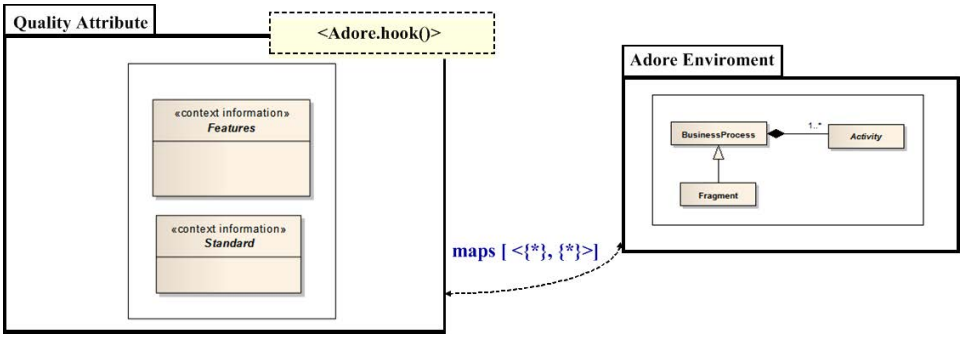

Figura 5.4: Propuesta de integración Theme/UML - **ADORE**.

La figura 5.5 presenta una aplicación de la propuesta de integración de las técnicas de modelado de comportamiento y multiples vistas, para formular un fragmento **ADORE** que representa el proceso de seguridad de control acceso bajo el estándar *XACML* definido en la sección 4.1.

La formulación del fragmento presentada en la Figura 5.5 permite eviden-

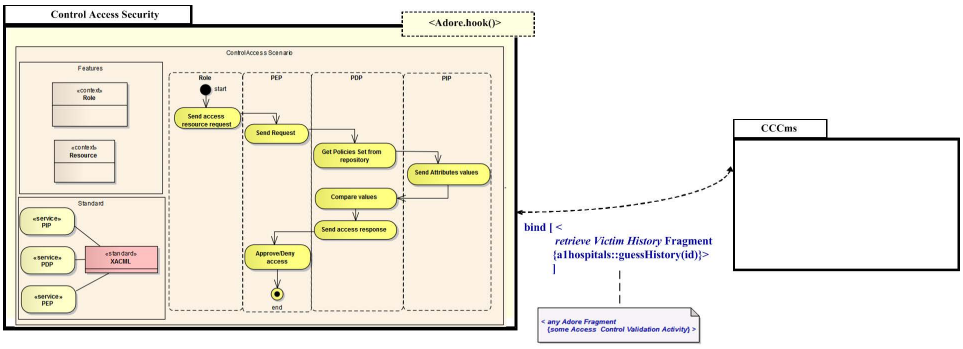

Figura 5.5: Generación de fragmento **ADORE** para soportar el estándar *XACML*.

ciar cómo dicho fragmento está definido en términos de una solución genérica (el fragmento está modelado desde el estándar *XACML*) más no del problema en sí mismo como se ha planteado en el fragmento de la Figura 5.3. La incorporación de *Theme/UML* permite definir el fragmento aspectual como un modelo genérico desde el punto de vista estructural y de comportamiento.

La sección 5.7 plantea la estandarización de comportamientos derivados desde atributos de calidad. La propuesta de integración de comportamientos (método **ADORE**) y estructura (modelos de múltiples vistas) complementa la propuesta formulada, incorporando el uso de *Theme/UML* con el fin de soportar la definición de modelos (fragmentos) genéricos enlazados posteriormente mediante la aplicación del método **ADORE**. El uso de técnicas manejadas por modelos permitiría que la definición de fragmentos genéricos sea considerada desde etapas tempranas de construcción del proceso de negocio, usando altos niveles de abstracción.

5.4. Definición de fragmentos ADORE genéricos para soportar seguridad

Todo el proceso de control de acceso *XACML*, presentado en la sección 4.1 se modela en un fragmento **ADORE**, dado que el proceso *XACML* se fundamenta en la invocación de servicios subyacentes que implementan las operaciones de seguridad. Así este fragmento puede ser posteriormente entrelazado a otros fragmentos y orquestaciones de servicios más grandes,

adaptando cualquier workflow mediante composición aspectual y la adición de este fragmento de control de acceso. La figura 5.6 presenta el fragmento **ADORE** para encapsular todo el proceso *XACML* en una unidad de modelado del proceso de negocio.

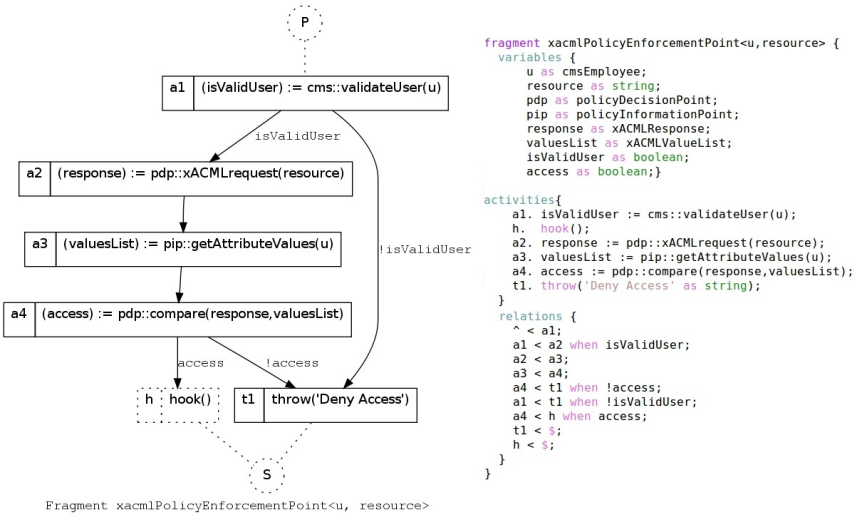

Figura 5.6: Fragmento **ADORE** XACML

El fragmento **ADORE** de seguridad para control de acceso basado en *XACML* es en sí mismo un *Policy Enforcement Point (PEP)*, de acuerdo al proceso solicitud-respuesta *XACML* definido en la sección 4.1. Dicho *PEP* captura la solicitud de acceso por parte del usuario, realiza la invocación al *PDP* encargado de obtener las políticas asociadas a la solicitud realizada, invoca al *PIP* para consultar los atributos propios de la política encontrada por el *PDP*, y finalmente compara dichos atributos con los atributos propios de la solicitud de acceso para proceder a permitir o denegar el acceso. Debe recordarse que el código presentado por el fragmento se encuentra comprometido en los servicios invocados de forma explícita en el fragmento, así que la posterior implementación del proceso de control de acceso *XACML* se deja bajo la responsabilidad de los expertos en seguridad.

Para fines de ilustrar el uso de fragmentos **ADORE** que aborden operaciones específicas de seguridad, a continuación se procede a proponer un fragmento genéricos para administrar el comportamiento propio de ope-

raciones de desencripción, usando el estándar *X.509* basado en *RSA*. La figura 5.10 presenta un fragmento **ADORE** para soportar la operación de desencripción basada en el algoritmo *RSA X.509*. El propósito es formular un segundo fragmento **ADORE** de seguridad encargado de gestionar la descripción de mensajes encriptados previamente bajo *RSA*, en el contexto de un proceso de negocio, mediante la invocación a servicios subyacentes que implementen dichas operaciones de encripción usando frameworks como *javax.crypto* en *Java*, *mcrypt module* en *PHP*, u otras tecnologías[1].

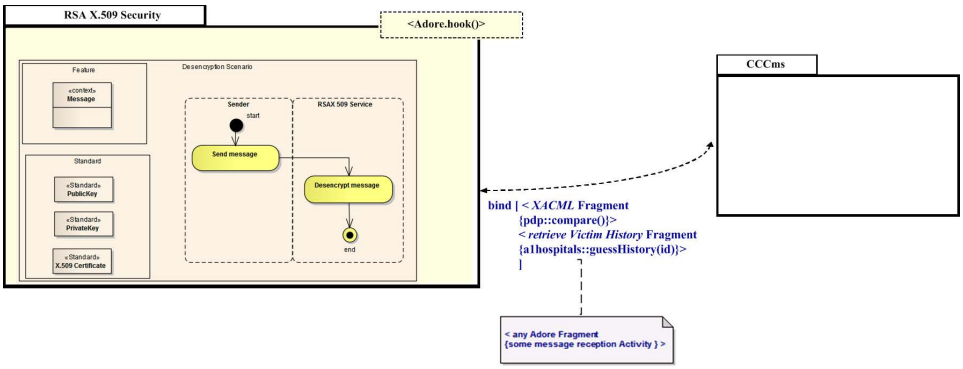

Figura 5.7: Generación de Fragmento **ADORE** para soportar*Desencripción RSA X.509*.

5.5. Aplicación de los fragmentos de seguridad en el *CCCms*

El caso de estudio sobre el cual se desarrolla este trabajo proviene de [46], quienes proponen un documento de especificación de requisitos, denominado *Crisis Management System - CMS*, para comparar diferentes técnicas de modelado orientadas a aspectos. De acuerdo con la definición dada en el caso de estudio, un *CMS* es un *sistema que facilita la coordinación de actividades y flujo de información entre los stakeholder y participantes que requieren trabajar conjuntamente para manejar una crisis.*

[1]ejemplo disponible en http://schneimi.wordpress.com/2008/11/25/aes-128bit-encryption-between-java-and-php/

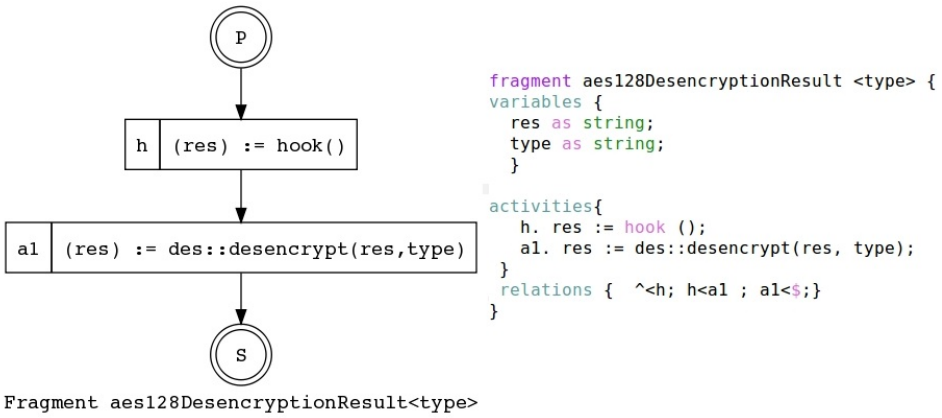

Figura 5.8: Fragmento **ADORE** para operación de *Desencripción RSA X.509*.

Debido a que el título del documento en sí mismo induce a pensar que se manejarán crisis de cualquier tipo (epidemias, ataques terroristas, desastres naturales, etc), los autores delimitan el contexto del documento de especificación de requisitos a accidentes de vehículos; por lo tanto, el *CCCms (Car Crash Crisis Management System)* incluye todas las funcionalidades de sistemas generales de gestión de crisis y algunas características particulares para *los accidentes de tránsito*, tales como *facilitar el rescate de las víctimas en el escenario de la crisis*, y *el uso de grúas para retirar vehículos dañados*.

El documento de especificación de requisitos define 10 casos de uso y 37 requisitos no funcionales correspondientes a los atributos de calidad de *Availability, Reliability, Persistence, Real-time, Security, Mobility, Statistic Logging, Multi-Access, Safety, Adaptability y Accuracy*[2]. Cada escenario de los casos de uso define en primer lugar un flujo básico (escenario exitoso) el cual representa el flujo normal de acciones para manejar una crisis (ejemplo: *recuperar la identidad de testigos, contactar los bomberos que se localizan más*

[2]Se opta por utilizar los nombres originales de los atributos de calidad en inglés para evitar confusiones con la traducción de los términos *security* y *safety*, ya que la traducción de ambos términos en español es seguridad, pero ambos tienen connotaciones diferentes acordes con la norma ISO 25010.3 SQuaRE: *security* se refiere al grado de protección de la información y datos, mientras que *safety* se refiere al el grado previsto de impacto por daño a personas, empresas, datos, software, bienes o al ambiente en contextos intencionados de uso.

cerca a la ubicación del accidente). Posteriormente, los casos de uso definen un conjunto de extensiones para considerar situaciones no abordadas por los flujos normales, por ejemplo, *los testigos ofrecen información falsa sobre un accidente* o *los bomberos no están disponibles para una intervención rápida.*

En el documento de especificación de requisitos para el *CCCms* se resalta que el *requisito de seguridad No 1* hace alusión a la definición de políticas de control de acceso donde éstas deben describir los componentes e información que los diferentes usuarios pueden adicionar, acceder y actualizar.

Con el fin de cumplir con el requisito No funcional de seguridad No 1 del documento de especificación de requisitos para el *CCCms* de [46](el cual expresa literalmente: *The system shall define access policies for various classes of users. The access policy shall describe the components and information each class may add, access and update - El sistema deberá definir las políticas de acceso para varios tipos de usuarios. Las políticas de control de acceso describirán los componentes y la información que cada tipo de usuario podría agregar, acceder y actualizar*), y de incorporar control de acceso a recursos altamente sensibles del *CCCms*, como las historias médicas de las víctimas (Caso de Uso No 7: **Execute Rescue Mission** - paso No 3 del flujo básico: *System requests victims medical history information from all connected HospitalResourceSystems. FirstAidWorker administers first aid procedures to victim - El sistema solicita información de la historia clínica de la víctima de todos los recursos de sistemas hospitalarios conectados. El rol FirstAidWorker administra el procedimiento de primeros auxilios a la víctima.*), se ha propuesto un fragmento **ADORE** encargado de administrar el acceso de un usuario (*FirstAidWorker*) a las historias médicas. La figura 5.6 presenta el fragmento **ADORE** para manejar el proceso solicitud-respuesta definido por el estándar *XACML*.

El fragmento *XACML* definido en las Figuras 5.5 y 5.6 puede ser entrelazado en otras actividades de otros fragmentos del *CCCms* que tengan invocaciones (llamadas) a servicios de hospitales con el fin de validar el control de acceso sobre una historia médica de una víctima por parte de algún rol; de esta forma puede apreciarse cómo el fragmento *XACML* encapsula y controla las acciones propias de la validación de control de acceso basadas en invocaciones de servicios subyacentes implementados para tal fin. Así la implementación del proceso y políticas basadas en *XACML* podría realizarse en cualquier ambiente tecnológico y/o de desarrollo, de tal forma que luego

se expongan las interfaces de los servicios para hacer las invocaciones respectivas desde los fragmentos **ADORE**.

Un ejemplo donde se puede exponer el entrelazado entre dos fragmentos se puede encontrar con el fragmento denominado *retrieveVictimHistory* elaborado para el *CCCms*. Dicho fragmento representa una extensión del caso de uso *Execute Rescue Mission*. El tercer paso del escenario principal de dicho caso de uso especifica literalmente: *System requests victims medical history information from all connected HospitalResourceSystems. FirstAidWorker administers first aid procedures to victim. - El sistema solicita la información de la historia médica de la víctima para todos los Sistemas de recursos hospitalarios. El role FirstAidWorker administra procedimientos de primeros auxilios a la víctima*. El fragmento **ADORE** para *retrieveVictimHistory* se presenta en la Figura 5.11. Dada la presencia del control de acceso en el fragmento *retrieveVictimHistory*, es posible entrelazar el fragmento *XACML* con el fragmento retrieveVictimHistory a través de la actividad **a1*hospitals::guessHistory(id)***. La figura 5.10 presenta el entrelazado de los fragmentos **ADORE** antes mencionados.

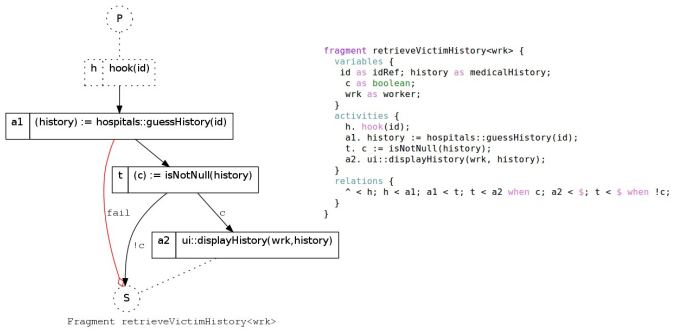

Figura 5.9: Fragmento **ADORE** *retrieve Victim History* definido para el *CCCms*.

Los fragmento entrelazados *XACML* y *retrieve Victim History* podrían entrelazarse con la actividad **a3 *hospitals::requestHistory(id)*** de la orquestación de servicios propuesta para el caso de uso *Execute Rescue Mission*. La Figura 5.11 presenta el entrelazado entre la orquestación *Execute Rescue Mission* y los fragmentos *XACML* y *retrieve Victim History* mediante la actividad **a3**.

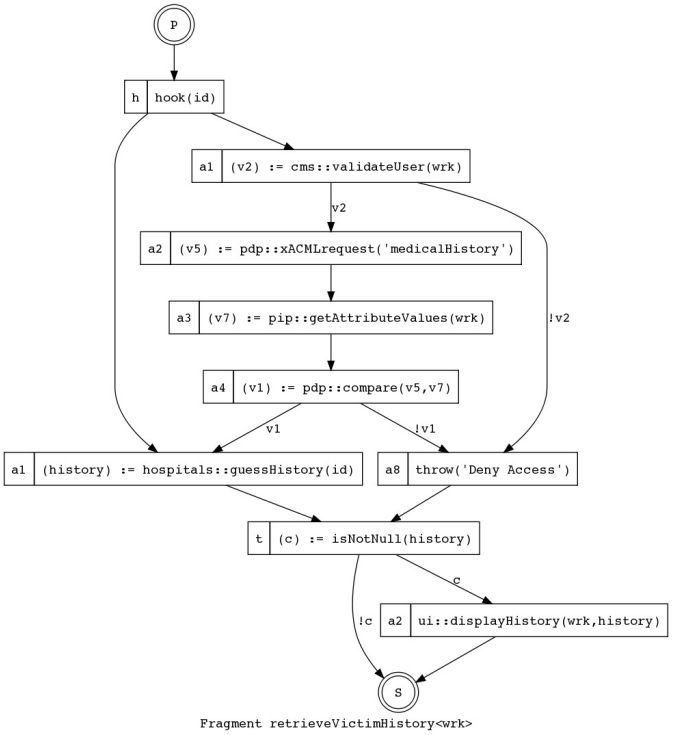

Figura 5.10: Entrelazado del Fragmento XACML sobre la actividad **a1** *hospitals::guessHistory(id)* del fragmento *retrieve Victim History*

Gracias al fragmento *XACML* previamente expuesto, es posible deducir un complejo contexto de intercambio de información crítica, en donde las *reglas de control de acceso* no serían controles suficientes para evitar apropiaciones mal intencionadas de la información que se está intercambiando entre los recursos hospitalarios a los cuales se conecta el *CCCms* para consultar la historia médica de una víctima, teniendo en cuenta que este tipo de información se considera altamente protegida debido a las consecuencias legales y éticas que se originaría por los accesos no autorizados. Trabajos como [34] y [65] proponen el reforzamiento y flexibilización de los mecanismos de *control de acceso* para incorporar operaciones de encripción de información intercambiada, de tal forma que sólo los usuarios con los privilegios de acceso correctos sean los encargados de desencriptar y accesar la información médica.

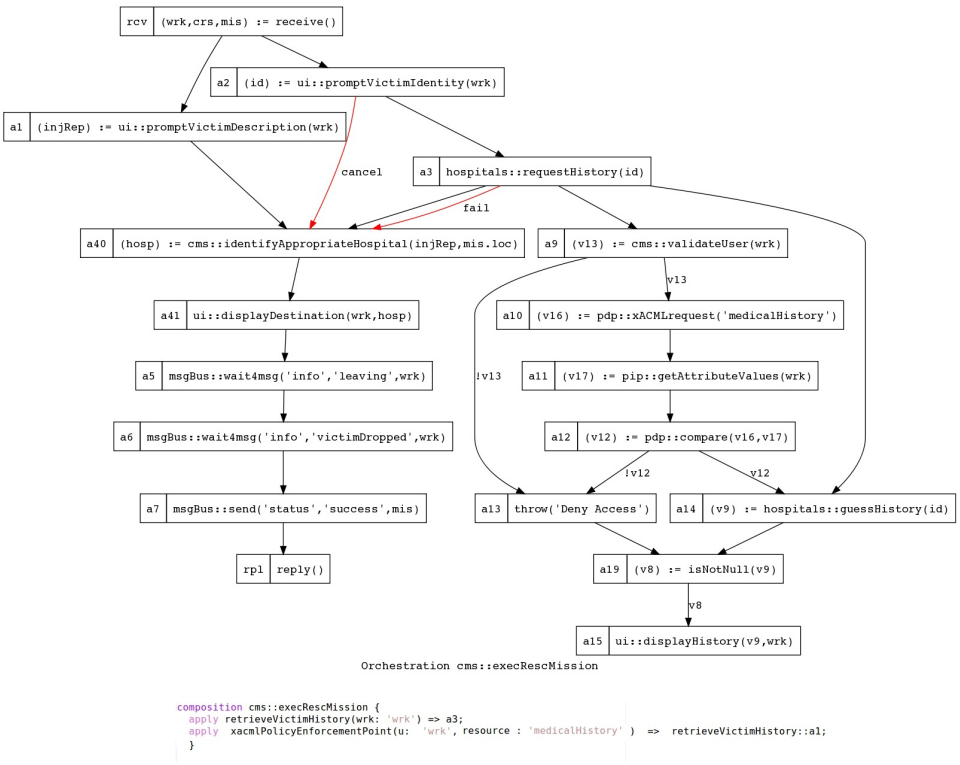

Figura 5.11: Entrelazado de fragmentos sobre la actividad **a3** de la orquestación *Execute Rescue Mission*

El requisito no funcional de seguridad No 3 para el *CCCms* - establece que: *All communications in the system shall use secure channels compliant with AES 128 standard encryption - Todas las comunicaciones en el sistema deberían u-sar canales seguros compatibles con el estándar de encripción AES 128.* Este requisito permite incorporar el fragmento de desencripción *RSA X.509* formulado anteriormente dentro del workflow del *CCCms*, con el fin de adicionar comportamientos para la gestión de información encriptada.

Al contemplar una estrategia de incorporación del fragmento *RSA X.509*, se evidencia que éste puede ser entrelazado en los fragmentos *XACML* y *retrieveVictimHistory*; para este caso, se ha entrelazado el fragmento *RSA X.509* con el fragmento *XACML* con el propósito de desencriptar el mensaje

(historia clínica de la víctima) que llega desde un recurso hospital, una vez que se haya establecido que un usuario tiene privilegios válidos de *control de acceso* sobre este recurso. La figura 5.12 presenta el entrelazado de los fragmentos *RSA X.509* y *XACML* a través de la actividad **a4** de este último (*pdp::compare()*); posteriormente, ambos fragmentos son entrelazados al fragmento *retrieveVictimHistory*, tal como se presenta en la Figura 5.13. Finalmente, la figura 5.14 presenta la orquestación *Execute Rescue Mission* con la adición de tres nuevos fragmentos para manejar comportamientos específicos (*control de acceso, desencripción* y *consulta de historias clínicas*).

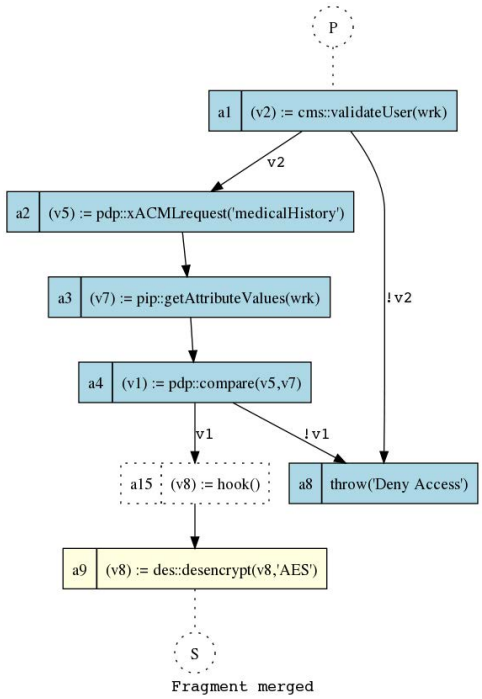

Figura 5.12: Entrelazado de los fragmentos *RSA X.509* y *XACML*

5.6. Trabajos previos *XACML* - workflows de procesos de negocio

Existen propuestas recientes para derivar políticas de control de acceso desde altos niveles de abstracción, las cuales combinan técnicas manejadas

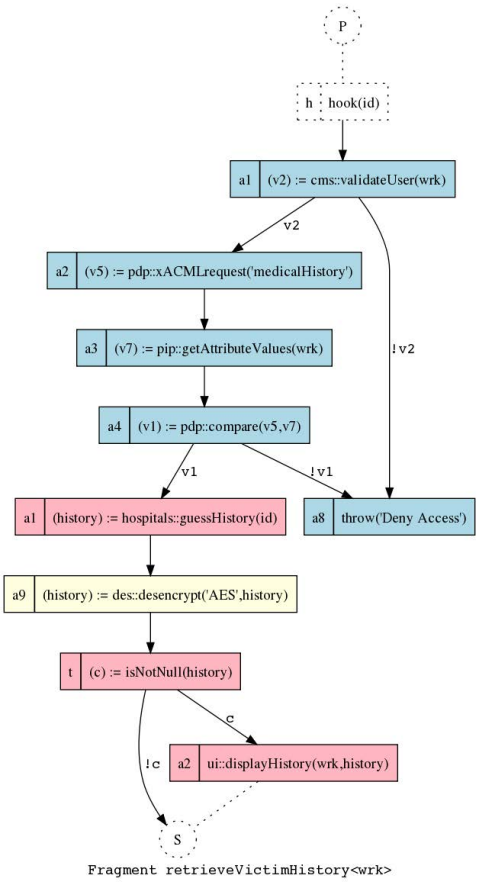

Figura 5.13: Entrelazado de los fragmentos *RSA X.509*, *XACML* y *retrieve-VictimHistory*.

por modelos y desarrollo orientado a aspectos. En [67] los autores exponen la formalización de un proceso de traducción de políticas de control de acceso en código de aspectos; los autores proponen el término *role slice* para representar un conjunto de métodos de clases sobre los cuales puede acceder un rol específico; el *role slice* representa el interés o concern que captura permisos para roles. En este trabajo, las políticas de control de acceso basadas en *role slice* (requisitos RBAC) son traducidos a código de programación orientada a aspecto.

Christiano Braga en [13] formula una técnica MDA para soportar la es-

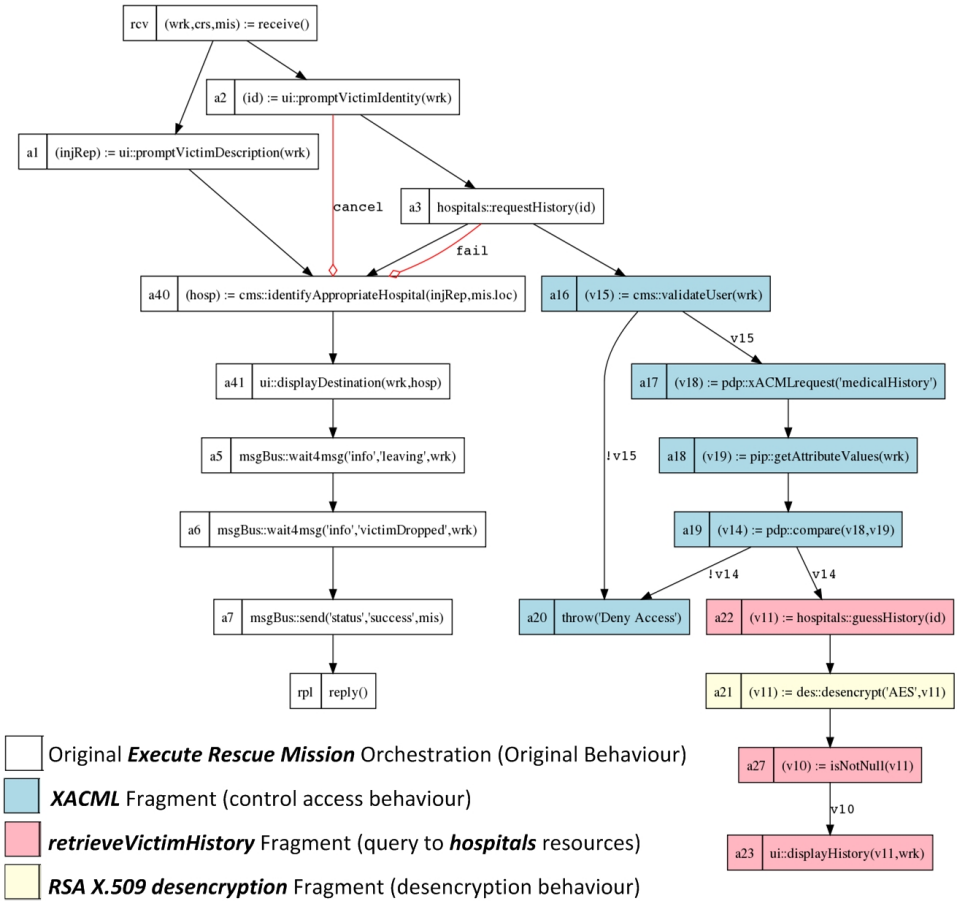

Figura 5.14: Orquestación *Execute Rescue Mission* resultante.

pecificación y validación de generación de código para políticas de control de acceso, apuntando hacia una infraestructura basada en aspectos. La propuesta de Braga contiene: *i)* un generador de código resultante de la transformación de modelos definidos en SecureUML, y *ii)* un lenguaje denominado *AAC - Aspects for Access Control*, lenguaje propuesto por el autor con el fin de mapear las transformaciones desde modelos *SecureUML*. En [68] los autores proponen políticas RBAC para web services usando una técnica MDA por capas junto con principios de modelado ágil de seguridad para mejorar requisitos de seguridad. Los autores usan métodos ágiles que soporta pruebas sobre los concerns o intereses a través de*pruebas de aceptación*. Las técnicas manejadas por modelos son usadas para especificar modelos que soporten la

integración de modelado ágil con actividades de seguridad.

Sin embargo, las propuestas anteriormente mencionadas no consideran la reutilización y orquestación de servicios que implementan estándares como *XACML* de *OASIS*, y su posterior aplicación en procesos de negocios. Adicionalmente, estas propuestas no consideran la encapsulación de comportamientos especiales derivados desde atributos de calidad como la *seguridad*, usando agrupaciones aspectuales de invocaciones de servicios. En lugar de ello, las propuestas están orientadas a la generación de código de aspectos desde políticas *RBAC* expresadas en modelos de clases que representan requisitos de control de acceso.

Gronmo propone en [41] el uso de transformaciones de grafos algebraicas para especificar aspectos *BPEL* a nivel de modelado, pero su trabajo no se centra en *concerns* o intereses derivados desde atributos de calidad como la seguridad. Los autores en [22] proponen una extensión orientada a aspectos para *BPMN* denominada *AO4BPMN*, bajo la motivación de incorporar conceptos orientados a aspectos en lenguajes para modelado de procesos de negocio; esta propuesta está dirigida a resolver la ausencia de mecanismos para expresar concerns en *BPMN* y lenguaje de modelado de procesos de negocio, pero *AO4BPMN* no contiene información sobre cómo entrelazar los aspectos dentro de un modelo base de procesos de negocio. En la propuesta presentada en esta sección se muestra el entrelazado entre fragmentos con el fin de adicionar nuevos comportamientos (*concerns*), derivados desde consideraciones de seguridad basada en *control de acceso*, dentro de un workflow de procesos de negocio previamente establecido.

El uso de *XACML* conjuntamente con modelos *BPM* ha sido previamente abordado en trabajos como [29] [81]. Dhankhar et al [29] formulan una técnica basada en el *paso de información contextual* para soportar el proceso de decisión de *control de acceso* dentro de un workflow controlado por políticas *XACML*; dicho workflow está construido mediante web services cooperativos. La información contextual se construye y verifica usando el framework *RDF Resource Description Framework*[3] propuesto por la *W3C*, y la coordinación entre servicios es realizada mediante *BPEL*. Wolter y Schaad [81] proponen un trabajo futuro relacionado con la derivación automática de políticas de autorización *XACML* a partir de una extensión para *BPMN* formulada por los mismos autores, la cual expresa autorizaciones dentro de un modelo de

[3] http://www.w3.org/RDF/

workflow, que habilita patrones de localización de recursos tales como separación de responsabilidades, localizaciones basadas en roles, manejo de casos o localizaciones basadas en históricos.

5.7. Estandarización de comportamientos derivados desde atributos de calidad

La formulación de fragmentos derivados desde operaciones de seguridad (*control de acceso* y *encriptación*) permite deducir la posibilidad de definir, en forma genérica, fragmentos u orquestaciones de comportamientos propios de atributos de calidad. Este tipo de fragmentos podrían ser reutilizados en diversos workflows de procesos de negocio, gracias a la separación de las invocaciones de servicios subyacentes agrupadas por los fragmentos de comportamiento específicos, del proceso de negocio abordado por los workflows. Este trabajo futuro contemplaría *i)* la definición a nivel del metamodelo **ADORE** de los comportamientos representativos de atributos de calidad con incidencia directa en la ejecución de workflows de procesos de negocio (ejemplo: *operaciones de seguridad como integridad, autenticación* y *confidencialidad*), y *ii)* la implementación mediante fragmentos **ADORE** de una biblioteca de comportamientos reutilizables en diferentes workflows.

Este trabajo futuro incluye también la adición un atributo de clasificación a los nuevos fragmentos formulados, de tal forma que dicho atributo representaría la categoría de atributo de calidad bajo la cual se formulan los fragmentos. De esta forma se facilitaría la administración de los fragmentos especialmente para fines de visualización de las composiciones y orquestaciones. Actualmente, la estructura del archivo XML generado desde **ADORE** para representar composiciones entre fragmentos y orquestaciones, no contempla etiquetas XML para identificar si un fragmento se deriva desde un atributo de calidad; por este motivo, si se requiere representar un patrón asociado a un atributo de calidad (por ejemplo, resaltar con un color específico los fragmentos que soportan la seguridad sección 6) dentro de una visualización de un workflow, es necesario indicar el nombre explícito de los fragmentos que administran operaciones relacionadas con este atributo.

Una rápida implementación de este trabajo futuro sería la incorporación de una etiqueta en el archivo XML generado por **ADORE** para las composiciones, junto con la modificación del importador XML desarrollado en *Moose*; sin embargo, este tag debería ser adicionado desde por el mismo motor de

ADORE, previo redefinción conceptual a nivel del metamodelo **ADORE**.

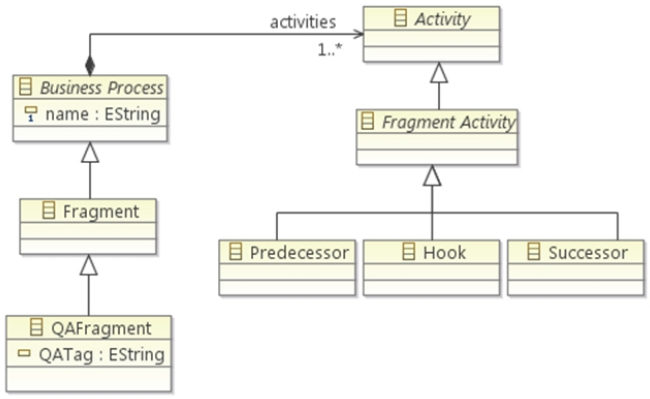

Figura 5.15: Propuesta modificación metamodelo **ADORE** para definir el concepto de *QAFragment*.

La figura 5.15 presenta una propuesta para enriquecer el metamodelo de **ADORE** definido en [58] con el fin de soportar la formulación de fragmentos o *procesos incompletos*, que administran atributos de calidad específicos mediante invocación de servicios especializados. Se propone la incorporación del concepto *QAFragment* al metamodelo **ADORE** que define el concepto de **Fragment**, mediante herencia entre la *EClass Fragment* y un elemento tipo *EClass*, *QAFragment* heredaría las características de un fragmento **ADORE**, y adicionalmente contendría un *EAttribute* tipo *EString* para identificar el tipo de atributo de calidad que está siendo manejado por el fragmento.

El atributo de clasificación (*Tag*) formulado para los fragmentos obtenidos desde atributos de calidad serviría inicialmente para tareas de visualización de fragmentos, de tal forma que la distinción de éstos se realizada mediante el *Tag* propuesto y no mediante el nombre de los fragmentos, tal como actualmente se debe implementar (sección 6).

La estandarización de comportamientos derivados desde atributos de calidad también implica un *análisis de variabilidad* con el fin de identificar y administrar adecuadamente las características variables de los fragmentos genéricos; éstas provienen desde la información del contexto de proceso de

negocio donde será aplicado el fragmento y el comportamiento de calidad a aplicar. En la Figura 5.3 se expuso un ejemplo de información de contexto de un fragmento **ADORE**. La administración de la *variabilidad* debe complementarse con los mecanismos ofrecidos por **ADORE** para soportar información de paso de parámetros que permiten generalizar los fragmentos dentro de workflows.

Capítulo 6

Visualización de fragmentos de seguridad en el *CCCms*

Tal como se mencionó anteriormente, la seguridad es quizás la aplicación más exitosa del principio de *AOSD*, teniendo en cuenta que este tipo de operaciones es altamente factible de encapsularse como fragmentos aspectuales que luego intervienen en ciertos procesos de negocio acorde a requisitos específicos.

La aplicación del principio de separación de intereses o concerns en contextos de workflows de procesos de negocio implícitamente deriva la presencia de la *complejidad* como factor importante a considerar dentro de las tareas de mantenibilidad, comprensibilidad y medición de la exactitud de un proceso de negocio. La medición de la complejidad en procesos de negocio se apoya en técnicas para control y seguimiento de la *complejidad de software*, especialmente técnicas basadas en **visualización de software**; ésta se define como *el uso de representaciones gráficas de aspectos o características del software como la estructura, comportamiento y evolución*[30]. La *estructura* se refiere a las partes estáticas y relaciones del sistema, particularmente partes que pueden ser inferidas sin ejecutar el software. Esta categoria incluye el código, estructuras de datos, invocaciones y organizaciones del programa en módulos. El *comportamiento* se refiere a la ejecución del software (secuencia de estados) con datos reales y/o abstractos. La *evolución* hace alusión al proceso de desarrollo de software, enfatizando el cambio del software sobre el tiempo para extender funcionalidades o remover defectos encontrados.

Mosser et al en[61] resalta una de las debilidades más sensibles de la técni-

ca *AOM* y separación de intereses:*la introducción de sobrecarga en el proceso de diseño*. Si bien los autores presentan como principal ventaja de la aplicación de separación de intereses la docilidad y flexibilidad introducidas en la administración de la complejidad, ellos también especifican que la sobrecarga en diseño no es visible en términos de resultados. Los autores presentan tendencias estadísticas basadas en análisis bidimensionales usando información de los fragmentos **ADORE**, acorde con consideraciones cognitivas previamente formuladas por Cardoso et al en [18][17]. Cardoso en [18], para fines de justificar la visualización de procesos de negocio, hace una analogía (mapeo) entre éstos y el software, de tal manera que se presenta cómo un proceso de negocio (modelado con un lenguaje como *BPEL* o **ADORE**) podría ser visto como un software que ha sido particionado en grupos de módulos que toman entradas y proveen salidas. La interacción entre estos módulos es precisamente especificada usando operadores lógicos predefinidos.

El proceso de visualización ayuda a la comprensión y mejora del subjeto de observación (software o proceso de negocio). El objetivo de la visualización es la ampliación de la *cognición* o adquisición y/o uso de conocimiento[74]. Caserta y Zendra en [20] presentan un compendio de métodos para visualización de características estáticas del software, resaltando representaciones 2D, bloques 3D, y metáforas de ciudades en 3D.

Como se ha presentado en la sección 5.4, **ADORE** provee funcionalidades para exportar los fragmentos y visualizar éstos como imágenes en formato PNG, sin embargo, la sobrecarga que esta representación trae consigo podría ocasionar problemas de legibilidad para comprender y asimilar procesos de negocio por parte de los diseñadores y arquitectos, especialmente cuando se derivan intereses desde atributos de calidad o inclusive intereses relacionados con conceptos de alto nivel de abstracción como aquellos derivados desde arquitecturas empresariales; en efecto, trabajos como [16] plantean la concepción de intereses que ejercen influencia sobre otros intereses de una arquitectura empresarial, y por ende, de los procesos de negocio como tal.

Vanderfeesten et al en [76] propone la adopción de principios que orientan el diseño de software, con el fin de definir y soportar métricas aplicadas en workflows procesos de negocio. Los principios empleados son: *la complejidad, la cohesión, el acoplamiento, la modularidad y el tamaño.*

ADORE permite extraer información desde la representación interna

de los procesos de negocio, de tal forma que es posible generar información relacionada con la estructura y métricas de los procesos de negocio. En [59] los autores han formulado con conjunto de visualizaciones para representar composiciones e identificar patrones y categorizaciones sobre fragmentos y orquestaciones **ADORE**, para lo cual combinan **ADORE** con **Mondrian**[1][51] con el fin de visualizar y evaluar gráficamente composiciones **ADORE**, usando la técnica de la *vista polimétrica*, que consiste en visualizaciones de dos dimensiones encargadas de mapear métricas de software previamente definidas por los diseñadores con atributos como tamaño, color, o posición.

Esta sección plantea la aplicación de técnicas de visualización previamente concebidas desde [59] para soportar la representación de fragmentos que mapean comportamientos derivados desde atributos de calidad (seguridad basada en control de acceso y encripción particularmente). Para este propósito se plantea el uso de algunas técnicas de visualización de software, basadas en las plataformas Mondrian y Moose[2], con el fin de ofrecer a los arquitectos y diseñadores de procesos de negocio una herramienta que les permita *i)* observar la ubicación de los fragmentos de seguridad formulados en el contexto del modelo global del *CCCms*,*ii)* Determinar la complejidad de los fragmentos y orquestaciones que forman el *CCCms*, incluyendo los fragmentos de seguridad formulados con sus respectivas incorporaciones (entrelazados) dentro de fragmentos u orquestaciones previamente establecidos,*iii)* visualizar la forma cómo se entrelazan los fragmentos de seguridad por medio de las actividades de los fragmentos y orquestaciones involucradas.

El código fuente del proyecto **ADORE** en Moose se encuentra en el enlace http://www.moosetechnology.org/tools/adore . Este proyecto permite generar visualizaciones a partir de objetos que importan archivos XML exportados por **ADORE**; dichos archivos contienen entre otras, información de métricas que el motor de **ADORE** calcula con base en el número de actividades ejecutadas, el tamaño del proceso en número de actividades, el número de rutas que un fragmento u orquestación contengan[59]. La figura 6.1 presenta un ejemplo de un archivo XML generado por el motor de **ADORE**, resaltando las métricas exportadas con base a la definición de las actividades del fragmento u orquestación.

[1] http://www.moosetechnology.org/tools/mondrian
[2] www.moosetechnology.org/

```
<process name="xacmlPolicyEnforcementPoint" fragment="true">
<categories>
  <isModifier value="false" />
  <isInhibitor value="false" />
  <isSuccessorsInhibitor value="false" />
  <isHandler value="false" />
  <isThrower value="false" />
</categories>

<activities total="5">
  <invoke>4</invoke>
  <reply>0</reply>
  <throw>1</throw>
  <misc>0</misc>
</activities>
<relations total="8">
  <wait_for>4</wait_for>
  <guard>4</guard>
  <weak_wait>0</weak_wait>
  <fail>0</fail>
</relations>
<complexity>
  <width>2</width>
  <height>7</height>
  <surface>14</surface>
  <maze>3</maze>
</complexity>
<usage>
  <apply>1</apply>
  <targets>1</targets>
</usage>
</process>
```

Figura 6.1: Ejemplo de XML generado por **ADORE** para representar métricas de fragmentos y orquestaciones.

Las Figuras 6.2 y 6.3 presentan dos visualizaciones tipo *radar*[3] generadas para el contexto del *CCCms*; la Figura 10 presenta una visualización en donde el alto y ancho de los diferentes elementos gráficos se calcula con base en el número de actividades invocadas y las longitudes de las posibles rutas dentro de cada orquestaci ón/fragmento. La Figura 6.3 visualiza los elementos del *CCCms* resaltando las actividades que hacen posible las conexiones entre las orquestaciones y fragmentos. Para efectos de representación gráfica, los fragmentos de negocio se representan en color **azul claro**, las orquestaciones en color **verde**, y los fragmentos adicionados para administración de operaciones de seguridad (control de acceso y desencripción) se representan en color **rojo**.

Ambas visualizaciones permiten identificar un patrón de aplicación de los fragmentos de seguridad en el contexto del *CCCms*. El color **azul claro**

[3]término seleccionado para representar las visualizaciones que permiten tener un panorama o resumen general de todo el workflow bajo consideración

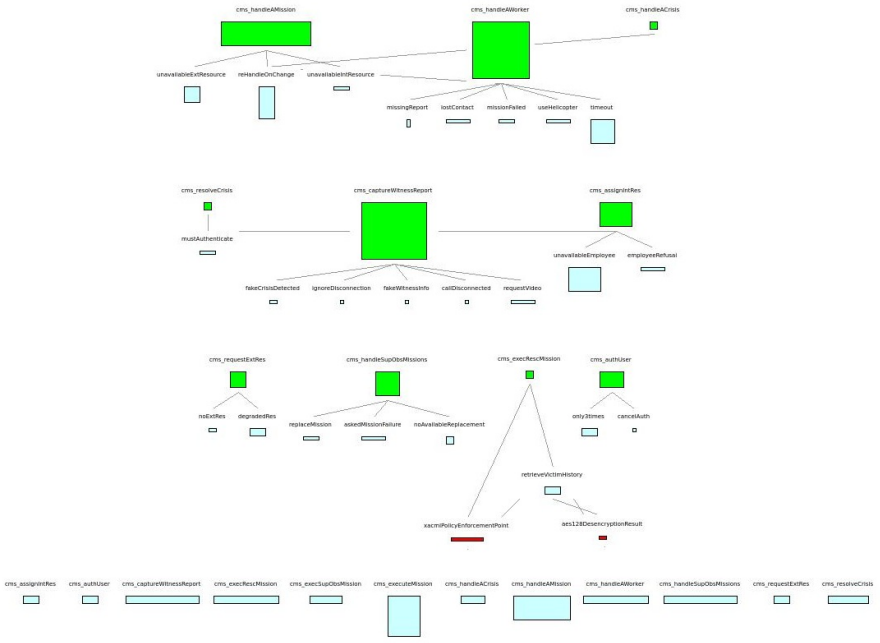

Figura 6.2: Complejidad de fragmentos y orquestaciones del *CCCms* incluyendo fragmentos de seguridad formulados.

aplicado a los fragmentos del proceso de negocio se propone con el fin de conservar el estandar empleado por herramientas CASE para coloreado de este tipo de artefactos de negocio (tipo *IBM Rational Suite*); el color **rojo** aplicado en los fragmentos de seguridad se propone por ser un color atrayente e impactante a primera vista, que a su vez sirve para representar un concepto crítico o un concepto con especial consideración. Derivado de la propuesta formulada en la sección 5.7, se desprende un trabajo de investigación futuro en el cual se analice *cuáles serían las formas de representación y colores más apropiados para visualizar los workflows de procesos de negocio, con sus respectivas adaptaciones derivadas desde atributos de calidad, de tal forma que los diseñadores y arquitectos de procesos distingan claramente las representaciones de la complejidad y las adaptaciones en cualquier workflow modelado con el método* **ADORE**. El código fuente para generar las visualizaciones expuestas se presenta en el apéndice B.

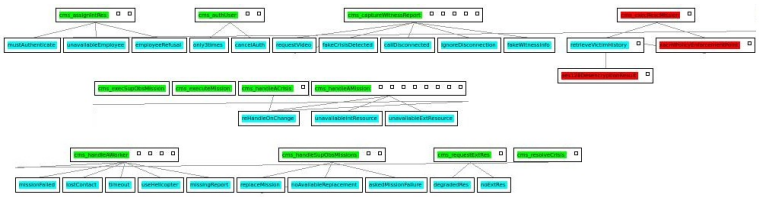

Figura 6.3: Actividades de conexión entre fragmentos y orquestaciones del *CCCms*, incluyendo fragmentos de seguridad formulados.

Capítulo 7

Conclusiones y perspectivas

7.1. Conclusiones de la propuesta desarrollada

Este trabajo ha expuesto una propuesta de integración de principios *MDD* y *ASOD* con el fin de establecer, a nivel de modelo, mecanismos de adaptación sobre workflows de procesos de negocio, con el propósito de incorporar restricciones de seguridad basada en *control de acceso* definido mediante el modelo *RBAC* y el estándar *XACML*, de tal forma que los workflows sean adaptados desde niveles de diseño. La propuesta hace uso intensivo del método **ADORE** (sección 3.1) para especificar tanto el proceso de negocio como las consideraciones de seguridad abordadas, las cuales son adicionadas al workflow mediante composición aspectual.

La propuesta presentada ha especificado un elemento composicional denominado *fragmento de orquestación* de acuerdo al método **ADORE**. Dicho fragmento se construye bajo los principios conceptuales establecidos en los metamodelos *SecureUML*(sección 4.4) y *XACML* (sección 4.1) para soportar operaciones de seguridad basada en control de acceso de roles. Posteriormente el fragmento se adiciona o entrelaza en un escenario de ejemplo definido en la sección 5.5. Finalmente se identifican métricas sobre workflows de acuerdo a la estructura de actividades del proceso de negocio, de tal forma que estas métricas permiten aplicar técnicas de visualización para evaluar tanto la complejidad del workflow como el impacto de las adaptaciones generadas por la incorporación de operaciones de seguridad. Esta propuesta propone el uso de técnicas de múltiples vistas basadas en *ThemeUML* para formular fragmentos **ADORE** con comportamientos genéricos derivados a partir de especificaciones de atributos de calidad manifestados como invocaciones de

servicios (sección 5.4).

El uso de operaciones de control de acceso basadas en *RBAC-XACML*, y las consideraciones formuladas en las secciones 5.2 y 5.7 plantean el enriquecimiento del método **ADORE** para incorporar seguridad y comportamientos derivados desde otros atributos de calidad. La formulación del fragmento de desencripción presentada en la sección 5.4, evidencia la posibilidad de construir procesos de negocio más robustos mediante la orquestación aspectual de fragmentos y las invocaciones de servicios soportadas por **ADORE**. La sección 6 expuso cómo las métricas definidas para procesos de negocio se pueden emplear para analizar la complejidad de los fragmentos que soportan operaciones de seguridad y control de acceso, de tal forma que las mismas se visualizan en el contexto general del workflow a nivel de modelo.

Con respecto a las preguntas de investigación formuladas en la sección 1.3, las secciones 5.3, 5.4 y 5.5 presentan la forma de aplicar y extender el método **ADORE** con el fin de considerar las operaciones de seguridad basada en el estándar *XACML* para control de acceso y *RSA X.509* para desencripción. La sección 5.1 plantea la administración de reglas de control de acceso bajo el enfoque aspectual y composicional de **ADORE**. La sección 5.2 presenta el aporte de los lenguajes de modelado basados en UML para representar la seguridad como un aspecto y sus posteriores intervenciones sobre una funcionalidad base definida en un caso de estudio. Finalmente el capítulo 6 presenta la aplicación de una técnica de visualización que permite analizar la complejidad de los nuevos fragmentos de seguridad formulados en este trabajo con base al caso de estudio empleado.

7.2. Conclusiones sobre el método ADORE

Esta propuesta expone la primera aplicación del método **ADORE** fuera del equipo original del proyecto **ADORE** perteneciente al grupo de investigación *MODALIS*, autores de dicho método. **ADORE** es un resultado de investigación muy reciente, y como tal demanda un trabajo contínuo para soportar las expectativas generadas por la administración aspectual de procesos de negocio a nivel de modelos. Como principal ventaja del método **ADORE** se resalta la relativa facilidad para abordar los workflows y sus adaptaciones desde altos niveles de abstracción, incorporando principios de *AOSD* para realizar las orquestaciones y composiciones de servicios. El esfuerzo invertido durante la formulación del método **ADORE** se refleja en la

simplicidad y agilidad con que se administran las orquestaciones de servicios y las adaptaciones de los procesos de negocio.

Otra ventaja comparativa de **ADORE** es su integración con entornos *Emacs*, el uso de *graphviz*[1] para la visualización de los workflows, y la previa implementación de funciones en *Prolog* para soportar la integración de fragmentos y la composición de las orquestaciones. El proyecto **ADORE** se basa en herramientas *open source*. Para este trabajo, el ambiente técnico del proyecto fue configurado en una máquina virtual montada bajo *Oracle VM VirtualBox*; en dicha máquina se instaló un sistema operativo *Ubuntu* Linux versión 9.0.

La ausencia de documentación sobre **ADORE** se convierte en la principal desventaja de este método; tanto la instalación como la interacción con **ADORE** requieren tareas técnicas de nivel intermedio o avanzado que no se encuentran oficialmente documentadas para la comunidad en general (ejemplo, FAQ de problemas de instalación de **ADORE** para la interacción con *graphviz* y *Prolog*). Adicionalmente, dado el poco tiempo que ha transcurrido desde el lanzamiento de la primera versión de **ADORE**, las instrucciones que se relacionan en el sitio oficial del proyecto para hacer composiciones de fragmentos son instrucciones netamente demostrativas; las composiciones de fragmentos deben ser implementadas posteriormente invocando funciones *Prolog* previamente programadas por los autores. Las implementaciones desarrollados para soportar **ADORE**, tanto a nivel de ejecución en ambientes Unix como el código fuente para visualización desplegado en *Moose* (sección 6), requieren esfuerzos permanentes para mejorar, incorporar nuevas funcionalidades, documentar y soportar el proyecto. También se hace necesario examinar la interacción con el menú **ADORE** que se incorpora en el editor *Emacs* una vez se ha concluido exitosamente la instalación. En el momento no se cuenta con un tutorial básico de interacción con el motor **ADORE** desde *Emacs*. Tampoco se cuenta con un tutorial para explicar las invocaciones de las funciones desarrolladas *Prolog* para realizar las composiciones de fragmentos a bajo nivel.

Uno de los trabajos más interesantes sobre **ADORE** (planteado en [58]) consiste en la aplicación de reglas de transformación **ADORE** - *BPEL*, con el fin de definir transformaciones genéricas entre los modelos aspectuales y el lenguaje estándar para automatización de procesos de negocio. También se

[1] http://www.graphviz.org/

hace necesario fortalecer los casos de aplicación de **ADORE**, de tal forma que la documentación oficial del proyecto relacione experiencias en entornos reales de administración de procesos de negocio. Los casos de estudio presentados contienen información valiosa sobre composiciones y orquestaciones aspectuales de servicios, sin embargo, se evidencia el origen académico de los mismos. La aplicación de **ADORE** sobre un escenario real de workflows seguramente impulsará la aplicación de este método a nivel industrial, convirtiéndolo en una alternativa competitiva frente a las diferentes extensiones formuladas para *WS-BPEL* y *BPEL* relacionadas en las secciones 2.4 y 5.6.

7.3. Trabajos futuros

7.3.1. Representación del contexto de ejecución del workflow (fragmentos de control)

La invocación de fragmentos aspectuales encargados de validar reglas de control de acceso puede generar una sobrecarga de procesamiento, debido a que las invocaciones de los servicios involucrados en un proceso basado en*XACML* (*PEP, PDP* y *PIP*), podrían repetirse para validar el acceso de un rol previamente evaluado. Por lo tanto, se propone la incorporación de fragmentos de control encargados de evitar la repetición de operaciones derivadas desde atributos de calidad que puedan afectar el comportamiento en ejecución del workflow. Dicho fragmento debería usar la información del contexto de ejecución del workflow en sí mismo, para lo cual se debería definir la forma de *representar* dicho contexto de ejecución, de tal forma que se tenga una instancia única que permitiera inclusive influenciar las operaciones de entrelazado de los fragmentos en un proceso de adaptación dinámica de workflows. Gracias a la orientación aspectual de los fragmentos en el método **ADORE**, se podría considerar un fragmento especializado para manejar la información del contexto del proceso de negocio, siendo éste enlazado en cualquier fragmento/orquestación que requiriera validar la ejecución previa de una operación similar. La pregunta de investigación generada a partir de esta consideración sería entonces *¿cómo representar el contexto de un proceso de negocio de tal forma que sea dinámicamente posible evitar la repetición de invocaciones previamente realizadas?.*

En la sección 5.3 se planteó el uso de *Theme/UML* con el fin de administrar la información relacionada al contexto de los fragmentos **ADORE** aplicado a la formulación de comportamientos genéricos. La representación

de contexto, además de usar fragmentos, podría considerar la aplicación de *Theme/UML* para derivar reglas y transformaciones que contribuyan a controlar las repeticiones de servicios previamente invocados.

7.3.2. Comparación del trabajo desarrollado con las propuestas BPEL4RBAC, AO4BPEL y AO4BPMN.

La sección 5.6 relaciona propuestas previas donde se contempla la incoporación de mecanismos especiales sobre *BPEL* para soportar *RBAC* y aspectos (*BPEL4RBAC y AO4BPEL* respectivamente); de igual forma también se expone una propuesta para soportar aspectos sobre la notación *BPM* (*AO4BPMN*). La propuestas de incorporación de extensiones sobre*BPEL* y *WS-BPEL* reflejan la falta de capacidad por parte de estos lenguajes para soportar comportamientos derivados desde atributos de calidad con impacto directo sobre la ejecución de los workflows de procesos de negocio.

Por otro lado, particularmente *AO4BPMN* no ofrece información sobre cómo entrelazar aspectos dentro de un modelo base. La propuesta de administración de fragmentos de seguridad basada en *RBAC-XACML* bajo el método **ADORE**, cubre las consideraciones planteadas por las propuestas anteriormente mencionadas, a nivel de modelo de procesos de negocio, sin recurrir a adiciones sobre un lenguaje o modificaciones sobre una notación [37].

En [72] se define un conjunto de características deseables en entornos de workflows adaptativos, entre las cuales se descatan:

- Adaptaciones basadas en customizaciones y optimizaciones que contemplan cambios en el contexto del workflow originados en la adición o reemplazo de servicios, y la detección/solución de cuellos de botella en el workflow.
- Estrategias predictivas de adaptación para evitar asignaciones de recursos que generen sobrecargas.
- Incorporación de un nivel de adaptación abstracto de workflow para definir estrategias de adaptación sin detalles de ejecución.
- Asignación de servicios a tareas específicas pertenecientes a instancias de workflows.
- Soporte para la definición de tareas reemplazantes en un workflow definidas antes de la ejecución de éste.

La propuesta de adaptación de workflows presentada en este trabajo cubre las características anteriormente expuestas, y adicionalmente abarca características no consideradas por las propuestas mencionadas en la sección 5.6, como el enlazado de intereses dentro de un modelo base, el reuso e incorporación de orquestaciones que implementan estándares concretos, y la encapsulación de comportamientos derivados desde atributos de calidad. Las propuestas de la sección 5.6 se resumen en la incorporación de aspectos y modelos de control de acceso *RBAC* en modelos *BPM*; ambas incorporaciones son realizadas en forma independiente. Este trabajo integra dichas incorporaciones para gestionar adaptaciones de workflows a nivel de modelo.

Se propone validar la aplicabilidad de las técnicas *BPEL4RBAC, AO4BPEL, AO4BPMN* y **ADORE** sobre un caso de estudio concreto que contemple workflows cuyas actividades demanden la incorporación de operaciones de seguridad. Idealmente dicha validación debería analizar el grado de facilidad con el cual los expertos en el proceso de negocio y en el control de acceso puedan formular e incorporar las adaptaciones sobre los workflows. La hipótesis de investigación a comprobar en este trabajo es si *la especificación de adaptaciones de workflows, a nivel de modelos, permite adaptar procesos de negocio en forma más eficiente que adaptaciones basadas en extensiones de lenguajes y notaciones sobre WS-BPEL y BPEL.*

7.3.3. Integración de propuestas y metamodelos

La sección 13.2 de [58] propone una técnica para mapear casos de uso orientados a aspectos (*AoUCM*[64]) con modelos de procesos de negocio del método **ADORE**, con el fin de permitir la encapsulación de intereses o *concerns* propios del nivel de requisitos, en artefactos de nivel de diseño. Los autores proponen un proceso iterativo, inspirado en metodologías ágiles, donde los artefactos de diseño e implementación interactuen conjuntamente para obtener el sistema final.

Dicha propuesta de integración de procesos se formula bajo un principio ágil soportado por la orientación a aspectos como el factor común de enlace entre ambos procesos. Por lo tanto, cualquier técnica aspectual que aborde una parte específica del proceso de desarrollo podría ser candidata para combinarse con el método **ADORE** y su respectivos artefactos de nivel de modelos, por ejemplo, la técnica de *Earyl Aspects* mencionada en la sección 2.1. De hecho, una interesante pregunta derivada de esta integración sería: *¿cuáles ventajas comparativas traería el uso del método* **ADORE** *para ad-*

ministrar fragmentos derivados desde atributos de calidad, con respecto a técnicas aspectuales como Early Aspects*[70] que emplean principios MDE con el fin de aplicar soluciones reusables que satisfacen requisitos no funcionales, y principios orientados a aspectos para mejorar la modularización de dichos intereses o concerns?*. Para resolver esta pregunta, y con el fin de validar el método **ADORE** respecto a técnicas de *Early Aspects*, se hace necesario ejecutar un trabajo futuro de automatización de la transformación definida en *Theme/UML* (sección 5.3), usando un enfoque de perfiles, para generar el código de fragmentos/orquestación definido en el modelo **ADORE**.

Por otra parte, la propuesta presentada expone la integración de tres metamodelos base: el metamodelo **ADORE**, el metamodelo *SecureUML* y el metamodelo *XACML*[2]. Dicho mapeo fue formulado para soportar conceptualmente la definición de reglas de control de acceso desde altos niveles de abstracción, de tal forma que los requisitos de seguridad basada en control de acceso establecidos para el *CCCms* pudiesen ser mapeados a nivel de modelo de igual forma a como **ADORE** modela los procesos de negocio. La integración propuesta entre metamodelos también consideró el mapeo de elementos genéricos *RBAC* definidos para el metamodelo *SecureUML*, en elementos del estándar *XACML* que materializan las políticas de control de acceso en ambientes de servicios distribuidos.

Las integraciones formuladas a nivel de procesos y de metamodelos deberían ser concebidas desde un marco de referencia de nivel superior que permita confrontar las múltiples vistas desde las cuales pueden apreciarse los artefactos generados, con respecto a los propósitos particulares de las iniciativas abordadas y las perspectivas generadas en el negocio mismo.

El uso de un marco de referencia permitiría formular reglas de transformación e integración entre procesos/metamodelos, aplicando un proceso sistémico para la definición y construcción de las diferentes integraciones y reglas de transformación. Por lo tanto, se propone un trabajo futuro basado en las siguientes preguntas de investigación:*i) ¿Es posible aplicar un marco de referencia para integrar métodos y técnicas basadas en principios AOSD/MDD, usadas para la administración de intereses o* **concerns** *desde altos niveles de abstracción?*, *ii) ¿Es posible aplicar un marco de referencia para integrar y transformar metamodelos que definen procesos de negocio con metamodelos donde se definen operaciones propias de atributos de calidad? iii) de*

[2]oficialmente no establecido como metamodelo por parte de OASIS.

acuerdo con el marco de referencia seleccionado, cuáles serían las etapas del proceso sistémico a aplicar para obtener la integración (y transformación) en cada uno de los casos anteriores?, iv) ¿Las propuestas de administración aspectual de intereses o concerns, a nivel de modelo, podrían convertirse en una ontología para adaptación de workflows de procesos de negocio?.

7.3.4. Uso de ADORE por partes de expertos de atributos de calidad (seguridad)

En [58] se plantean futuras pruebas para **ADORE** basadas en el modelado de aplicaciones previamente planteadas (como el caso de estudio *CCCms*[3], de tal forma que los diseñadores de procesos de negocio son consultados sobre el modelado de procesos usando herramientas comunes y usando el método **ADORE**. Un futuro trabajo que cobra especial importancia para la definición de fragmentos derivados desde atributos de calidad, es la validación del método **ADORE** por parte de expertos en estos dominios, por ejemplo, expertos en control de acceso basado en roles o expertos en encriptación.

Este trabajo futuro validaría si *¿las extensiones propuestas en este trabajo, definiendo fragmentos genéricos de seguridad, son útiles y permiten a los expertos de seguridad reducir el esfuerzo en el abordaje del atributo de calidad de la seguridad, evidenciando así una mejora con respecto al tratamiento tradicional de los atributos de calidad y su posterior incorporación en workflows?.*

7.3.5. Transformaciones ADORE - Moteur

Una alternativa para materializar los modelos **ADORE** consiste en definir reglas de transformación de dichos modelos hacia plataformas específicas de manejadores de workflows como *Moteur*[39][38]. Este manejador de workflows soporta técnicas basadas en SOA para proveer máxima flexibilidad en el reuso y ensamblaje de código a ser ejecutado sobre infraestructuras de Grid computing. Adicionalmente, *Moteur* ofrece estrategias de optimización para expresar aplicaciones complejas donde se manipulan grandes cantidades de datos [57]. La figura 5.15 presenta una aproximación inicial para proponer un mapeo semántico entre un fragmento **ADORE** y un fragmento del lenguaje *Gwendia* sobre el cual se definen los workflows a ser ejecutados en

[3]Más casos de estudio de **ADORE** disponibles en la sección *Examples/Case Study* de http://www.adore-design.org/doku/

Moteur.

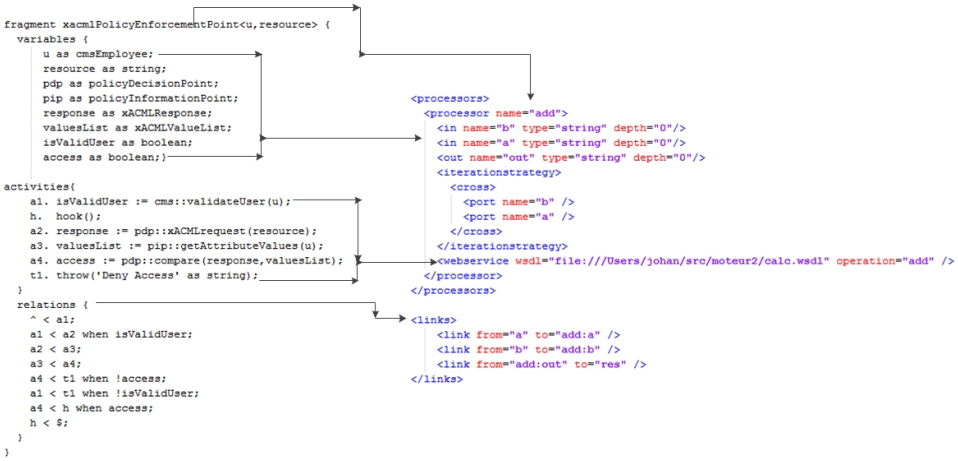

Figura 7.1: Consideraciones iniciales para mapeo **ADORE**-*Moteur*.

Este trabajo futuro se propone bajo la siguiente pregunta de investigación: *¿es posible definir a nivel semántico reglas de transformación que permitan convertir modelos* **ADORE** *en elementos* Gwendia *para ser ejecutados sobre el workflow* Moteur*, considerando el proceso de optimización de datos propuesto por este manejador de workflow?*.

Parte I

Apéndices

Apéndice A

Código de fragmentos de seguridad

A.1. Código ADORE del fragmento XACML

```
fragment xacmlPolicyEnforcementPoint <u ,resource> {
variables {
    u as cmsEmployee;
    resource as string;
    pdp as policyDecisionPoint;
    pip as policyInformationPoint;
    response as xACMLResponse;
    valuesList as xACMLValueList;
    isValidUser as boolean;
    access as boolean;}

activities{
    a1. isValidUser := cms::validateUser(u);
    h.  hook();
    a2. response := pdp::xACMLrequest(resource);
    a3. valuesList := pip::getAttributeValues(u);
    a4. access := pdp::compare(response,valuesList);
    t1. throw('Deny Access' as string);
  }
  relations {
    ^ < a1;
    a1 < a2 when isValidUser;
```

```
    a2 < a3;
    a3 < a4;
    a4 < t1 when !access;
    a1 < t1 when !isValidUser;
    a4 < h when access;
    h < $;
  }
}

depict xacmlPolicyEnforcementPoint using 'lightblue';
```

A.2. Código ADORE del fragmento de desencripción

```
fragment aes128DesencryptionResult <type> {
variables {
  res as string;
  type as string;
  }

activities{
   h. res := hook ();
   a1. res := des::desencrypt(res, type);
 }
 relations {  ^<h; h<a1 ; a1<$;}
}

depict aes128DesencryptionResult using 'lightyellow';
```

A.3. Consideraciones para la composición de fragmentos

Desde la perspectiva del método **ADORE**, las composiciones manejadas por modelos requieren el uso de lenguajes de dominio específico para soportar los mecanismos de composición propios del enfoque orientado a aspectos.

Un trabajo similar se encuentra en [8], donde los autores especifican un DSL para especificar un profile UML que permite adaptar diagramas de actividades con el fin de representar la composición sistemática de workflows a partir de servicios que se encuentran en una Grid. Para efectos del presente trabajo se presentarán los fragmentos y composiciones a nivel de modelos.

Actualmente se está trabajando en proporcionar mayor semántica a las composiciones en **ADORE**. El método **ADORE** propone la especificación de las composiciones de fragmentos usando un código similar al que se expone a continuación:

```
require 'processes/execRescMission.adore';
require 'fragments/retrieveVictimHistory.adore';
require 'fragments/xacmlPolicyEnforcementPoint.adore';

composition cms::execRescMission {
  apply retrieveVictimHistory(wrk: 'wrk') => a3;
  apply  xacmlPolicyEnforcementPoint(u:  'wrk', resource :
 'medicalHistory')  =>  retrieveVictimHistory::a1;
}
```

Sin embargo, la definición de estas composiciones aún no se encuentra plenamente soportada por el motor de **ADORE**. Por tal motivo, se hace necesario aplicar funciones *Prolog* desarrolladas por los autores de **ADORE** para realizar el proceso de composición de fragmentos y generación de las imágenes de los orquestaciones yfragmentos enlazados.

A continuación se transcribe un mensaje de correo electrónico enviado por *Sébastien Mosser* (autor de **ADORE**) donde se presentan las instrucciones para invocar las funciones implementadas en *Prolog*. Se decide presentar el mensaje electrónico original recibido con el fin de replicar éste de tal forma que sirva como tutorial mientras se implementa la traducción del código DSL **ADORE** a las respectivas composiciones. La sección A.4 presenta el código *Prolog* aplicado en esta propuesta.

```
2010/10/5 Sebastien Mosser <sebastien.mosser@gmail.com>

    Hi Faber,
```

```
the following code:

> composition cms::captureWitnessReport {
> apply callDisconnected => a10; // 1a.
> apply callDisconnected => a2; // 2a.
> apply requestVideo(user: 'coord') => {a3,a4}; // 3a.
> apply ignoreDisconnection => a4; // 4a.
> apply fakeWitnessInfo => a2a3; // 5a.
> apply fakeCrisisDetected => a4; // 5b.
> apply fakeCrisisDetected => requestVideo::a3; // 5b.
> }

is only "demonstrative" for now. the algorithm scheduler used
to translate the DSL into the associated set of composition
algorithm calls is highly experimental, and I have no time to
 make it more "robust" right now.

As far as we are, composition are described manually to the
engine.  I guess you'll use the four main ADORE algorithm:
clone (or instantiate),  merge and weave.

The doCompo.adore file in the "cccms/cosmetic" directory
 contains example  of code associated to the CCCMS
 example.
 ==>> http://code.google.com/p/adore/source/browse/
  trunk/case_studies/cccms/cosmetic/doCompo.adore

####
## Prerequisite
####

The composition are explained in Plain Old Prolog. Consequently,
 you'll have to  "declutch" the ADORE compiler. this is done with
 the use of /*% ... %*/ comment  in an ADORE file:

ADORE DSL
/*%
 Plain old prolog
%*/
```

```
ADORE DSL

I'll not assume any Prolog knowledge or deep understanding.
However, you **MUST**  be aware that:
 - symbols starting by an upper case letters are considered as
   free variable in Prolog,  and will be treated differently. You
   should not use such symbols.
 - Prolog is declarative, and there is no "output" values or others
   weird things in its syntax.  clauses are expressed sequentially,
   comma-separated.

Another thing: weave and merge "destruct" their inputs. You'll
need to clone (or instantiate)  the input BEFORE using it.

####
## Clone
####

The algorithm clone a fragment into a new one. The syntax is
 quite simple:

/*%
doClone(origin, cloneId),
%*/

After the execution of this clause, cloneId contains a clone
of 'origin'.

####
## Instantiate
####

Instantiate is an extension of "clone". It allow you to use
template parameters, syntactically  replaced while cloned.

/*%
doInstantiate(origin, target, cloneId, [bind(tpl,real), ... ]),
%*/

Based on an "origin" fragment and a "target" process (a fragment
```

```
or an orchestration), the clause will generate a fragment named
"cloneId", where all usage of "tpl" are replaced by "real".  "real"
 should be a variable defined in "target", and "tpl" **MUST** be
 expressed as a template  parameter in the DSL (syntax with '<...>').

%%%%
%% Merge
%%%%

Merge basically ... merge a set of fragment into a merged one.

/*%
doMerge([f1,...,fN], merged),
%*/

merged is the reult of the merge of the set of fragment {f1 ... fN}.


%%%%
%% Weave
%%%%

Weave integrates several fragments into an orchestration, in //.

/*%
 doWeave([weave(fragment, [act1,....,actN]), ... ])
%*/

Fragment is woven on the target block defined by the {act1, ... actN}
set.

%%%%
%% Example
%%%%

The Capture witness report composition is then describe manually as the
following  (and will be one day automatically transformed in a robust way ...

/*%
 doInstantiate(requestVideo, cms_captureWitnessReport, uc22_rV_1,
```

```
              [bind(user,coord)]),
 doClone(fakeCrisisDetected,uc22_fCD_1),
 identifyClone(requestVideo_a3,uc22_rV_1,UC22_rva3),
 doWeave([weave(uc22_fCD_1,[UC22_rva3])]),
 doProcessSimplification(uc22_rV_1),
 cccms_draw(uc22_rV_1,'business-only/2.2_requestVideoEnhanced'),

 doClone(callDisconnected, uc22_cD_1),
 doClone(callDisconnected, uc22_cD_2),
 doClone(fakeWitnessInfo,uc22_fWI_1),
 doClone(ignoreDisconnection,uc22_iD_1),
 doClone(fakeCrisisDetected, uc22_fCD_2),
 doMerge([uc22_iD_1,uc22_fCD_2], uc22_iD_fCD_1),
 cccms_draw(uc22_iD_fCD_1,'business-only/2.2_mergedOnA4'),

 doWeave([ weave(uc22_cD_1,     [ cms_captureWitnessReport_a10  ]),
           weave(uc22_cD_2,     [ cms_captureWitnessReport_a2   ]),
           weave(uc22_rV_1,     [ cms_captureWitnessReport_a3,
                                  cms_captureWitnessReport_a4   ]),
           weave(uc22_iD_fCD_1, [ cms_captureWitnessReport_a4   ]),
           weave(uc22_fWI_1,    [ cms_captureWitnessReport_a2a3 ])]),
 doProcessSimplification(cms_captureWitnessReport),
 cccms_draw(cms_captureWitnessReport,'business-only/2.2.captureWitnessReport'),
%*/

There are 3 clause I haven't explained in my previous example:

 - cccms_draw: a special clause defined for the CCCMS, whiwh produces
   a PNG output for the given input, programatically.

 - identifyClone: since "clone" rename all variables and activities, one need
   to use this predicate to identify the new name of an entity in a clone. In
   line 4, I'm looking for the  new name of the activity requestVideo_a3 in the
   cloned process uc22_rV_1. The predicate will unify the new name with UC22_rva3
  (notice the UPPER CASE first letter, indicating to Prolog that the variable is
  free and must be unified).

 - doProcessSimplification: identify several redundant relations and retract them.
    The  algorithm is NP-Complete, wo a large graph may consumes a lot of time to
    be simplified, that's why it is not automatic (typically more than 10 minutes for
```

```
        the "handleAWorker" use case.


    ####
    ## Reminder
    ####

    Identifiers are generated by the compiler to be unique.

     - process: the process service::operation is identified in the engine
       as service_operation.
     - fragment: name is assumed unique, so the fragment 'f' is
        identified as 'f'
     - activities and variables: these elements are prefixed by their
        container. the  activity 'act'  of the fragment 'frag' is then
        identified as 'frag_act'. Variables follow  the same scheme.
       The variable 'var', defined in the orchestration 'service::operation'
        will be identified  as 'service_operation_var'

    Cheers,

    --
     Sebastien
```

A.4. Código fuente de composiciones usando funciones implementadas en Prolog

A continuación se presenta el código fuente en *Prolog* de las composiciones de fragmentos relacionados en la sección 6, de acuerdo a la explicación expuesta en la sección A.3:

```
doSeqComposition :-
   doInstantiate(aes128DesencryptionResult, retrieveVictimHistory, ucvh,
[ bind(type, 'AES')]),
   doWeave([weave(ucvh,[retrieveVictimHistory_a1 ])]),
   doProcessSimplification(retrieveVictimHistory),
   cccms_draw(retrieveVictimHistory,'DesEncrypt_RH'),
```

```
  doInstantiate(xacmlPolicyEnforcementPoint, retrieveVictimHistory, xpe,
  [bind(u,wrk),bind(resource,medicalHistory)]),
  doWeave([weave(xpe,[retrieveVictimHistory_a1 ])]),
   doProcessSimplification(retrieveVictimHistory),
   cccms_draw(retrieveVictimHistory,'secure_retrieveVictimeHistory'),


  doInstantiate(retrieveVictimHistory, cms_execRescMission, rvh,
  [bind(wrk,wrk)]),
  doWeave([weave(rvh,[cms_execRescMission_a3 ])]),
  doProcessSimplification(cms_execRescMission),
  cccms_draw(cms_execRescMission,'secure_cms_execRescMission').



doComposition :-
  doInstantiate(xacmlPolicyEnforcementPoint, retrieveVictimHistory, xpe,
  [bind(u,wrk),bind(resource,medicalHistory)]),

  doInstantiate(aes128DesencryptionResult, retrieveVictimHistory, ucvh,
[ bind(type, 'AES')]),
   doMerge([xpe,ucvh], merged),
   doProcessSimplification(merged),
   cccms_draw(merged,'xamlAndDesEncrypt'),

   cccms_draw(retrieveVictimHistory,'retrieveVictimHistory'),
   doWeave([weave(merged,[retrieveVictimHistory_a1 ])]),
   doProcessSimplification(retrieveVictimHistory),
   cccms_draw(retrieveVictimHistory,'desencrypted_victim_history'),

  cccms_draw(retrieveVictimHistory,'secure_retrieveVictimHistory'),
  doInstantiate(retrieveVictimHistory, cms_execRescMission, rvh,
  [bind(wrk,wrk)]),
  doWeave([weave(rvh,[cms_execRescMission_a3 ])]),
  doProcessSimplification(cms_execRescMission),
  cccms_draw(cms_execRescMission,'secure_cms_execRescMissionSet').


shouldDraw(true).    %% Draw resulting process as PNG file
```

```
% shouldDraw(false). %% Do not draw resulting process as PNG file


shouldDraw(true).    %% Draw resulting process as PNG file
% shouldDraw(false). %% Do not draw resulting process as PNG file

cccms_draw(_,_) :- shouldDraw(false),!.
cccms_draw(Process,PrettyName) :-
swritef(RealFile,'output/%w.png',[PrettyName]),
        adore2png(Process,RealFile).
```

La ejecución de este código se debe realizar tal como se indica a continuación:

1. Almacenar las instrucciones presentadas anteriormente en un archivo *Prolog* con extensión *pl*, para este caso, *composition.pl* .

2. Ejecutar el motor **ADORE** e iniciar una sesión interactiva (opción *Adore - Adore Engine - Run Interactive Session* de *Emacs*), tal como se presenta en la figura A.1.

3. en la consola desplegada, escribir

   ```
   ?- consult('composition.pl').
   ```

   Aparecerá un mensaje similar a

   ```
   % composition.pl compiled 0.00 sec, 1,728 bytes
   ```

4. Ejecutar las instrucciones de composición escribiendo en la consola

   ```
   ?- doComposition.
   ```

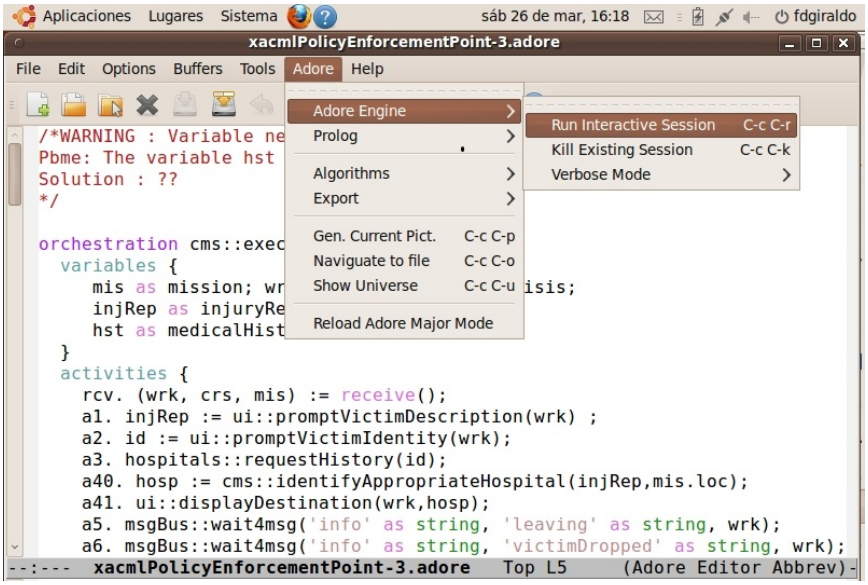

Figura A.1: Apertura de una sesión interactiva en **ADORE**.

Una vez ejecutado este comando se generan las imágenes en formato *PNG* correspondientes a la composición de la orquestación y los fragmentos previamente relacionados. Debe tenerse en cuenta que este ap éndice expone el código de los fragmentos **ADORE** para las operaciones de seguridad (*XACML* y desencripción). Los códigos del fragmento *retrieve Victim History* y la orquestación *Execute Rescue Mission* del *CCCms* se conservan tal como fueron definidos en [61].

Apéndice B

Código fuente de visualizaciones

Las visualizaciones presentadas en la sección 6 se generan en *workspaces* de *Moose*. Esto implica que previamente se ha descargado una copia del proyecto **ADORE** en la imagen local de *Moose* desde el repositorio *Monticello*[1]. Para ejecutar las visualizaciones relacionadas se deben ejecutar los siguientes pasos:

1. Estando en *Moose*, seleccionar la el paquete **Adore-Test**, luego seleccionar la clase **AdoreTest** y seleccionar el método **taosdBeforeV11**.

2. Estando en el método *taosdBeforeV11*, se cambia el nombre del método por **taosdWithSecureFragments**. Dicha acción crea un nuevo método que a su vez es una copia de*taosdBeforeV11*.

3. Adicionar en el método *taosdWithSecureFragments* recién creado, las siguientes líneas de código XML, en el bloque de definición de procesos:

```
<process name="xacmlPolicyEnforcementPoint" fragment="true">
 <categories>
   <isModifier value="false" />
   <isInhibitor value="false" />
   <isSuccessorsInhibitor value="false" />
   <isHandler value="false" />
```

[1] más instrucciones en http://www.moosetechnology.org/tools/Adore.

```
        <isThrower value="false" />
      </categories>

  <activities total="5">
        <invoke>4</invoke>
        <reply>0</reply>
        <throw>1</throw>
        <misc>0</misc>
      </activities>
      <relations total="8">
        <wait_for>4</wait_for>
        <guard>4</guard>
        <weak_wait>0</weak_wait>
        <fail>0</fail>
      </relations>
      <complexity>
        <width>2</width>
        <height>7</height>
        <surface>14</surface>
        <maze>3</maze>
      </complexity>
      <usage>
        <apply>1</apply>
        <targets>1</targets>
      </usage>
    </process>

     <process name="aes128DesencryptionResult" fragment="true">
      <categories>
        <isModifier value="false" />
        <isInhibitor value="false" />
        <isSuccessorsInhibitor value="false" />
        <isHandler value="false" />
        <isThrower value="false" />
      </categories>

      <activities total="1">
        <invoke>1</invoke>
        <reply>0</reply>
        <throw>0</throw>
```

```
    <misc>0</misc>
  </activities>
  <relations total="3">
    <wait_for>3</wait_for>
    <guard>0</guard>
    <weak_wait>0</weak_wait>
    <fail>0</fail>
  </relations>
  <complexity>
    <width>1</width>
    <height>4</height>
    <surface>4</surface>
    <maze>1</maze>
  </complexity>
  <usage>
    <apply>0</apply>
    <targets>0</targets>
  </usage>
</process>
```

El bloque XML que contiene los tags *isModifier, isInhibitor, isSuccessorsInhibitor, isHandler* y *isThrower* debe ser copiado de un bloque **process** previo, ya que actualmente **ADORE** no genera esta información. Los bloques *activities, relations, complexity, usage* y *targets* son generados automáticamente por **ADORE**.

4. Adicionar manualmente los siguientes bloques XML que se encargan de mapear las composiciones entre los fragmentos formulados para esta propuesta. Dichos bloques *composition* tienen números consecutivos que representan los entrelazados entre fragmentos:

```
<composition id="context_150" target="cms_execRescMission" >
    <apply id="apply_152" fragment="xacmlPolicyEnforcementPoint">

      <block id="block_151" process="cms_execRescMission">
        <activityRef uid="a3" />
      </block>
    </apply>
```

```
</composition>

<composition id="context_153" target="xacmlPolicyEnforcementPoint" >
    <apply id="apply_155" fragment="retrieveVictimHistory">

      <block id="block_154" process="xacmlPolicyEnforcementPoint">
        <activityRef uid="a4a4" />
      </block>
    </apply>
</composition>

<composition id="context_156" target="retrieveVictimHistory" >
    <apply id="apply_158" fragment="aes128DesencryptionResult">

      <block id="block_157" process="retrieveVictimHistory">
        <activityRef uid="a1" />
      </block>
    </apply>
</composition>

<composition id="context_159" target="aes128DesencryptionResult" >
    <apply id="apply_161" fragment="retrieveVictimHistory">

      <block id="block_160" process="aes128DesencryptionResult">
        <activityRef uid="a1" />
      </block>
    </apply>
</composition>
```

Una vez modificado el archivo XML, con la adición de la información de los fragmentos y composiciones de este trabajo, se procede a implementar las visualizaciones tal como se expone en la siguiente sección.

B.1. Código visualización Figura 6.2

El código que se relaciona a continuación corresponde al método denominado **viewCCCmTree**, que se encuentra en la clase **AdoreUniverse** del paquete **Adore-Core**. Este método se encuentra clasificado en la categoría

visualization de la clase **AdoreUniverse**.

```
viewCCCmsTree

| view |
view := MOViewRenderer new.
view shape rectangle withoutBorder.
view nodes: compositions forEach: [:each |
view shape rectangle withoutBorder; text: #compositionTargetName.
view interaction forwarder.
view node: each.
view shape rectangle
fillColor: Color green;
width: [:c | c numberOfTargetedActitivies * 10];
height: [ : c | c numberOfApplications * 10].
view interaction forwarder.
view node: each.
view verticalLineLayout center.
].

view shape rectangle withoutBorder.
view nodes: processes forEach: [:each |
view shape rectangle withoutBorder; text: #processName.
view interaction forwarder.
view node: each.
view shape rectangle
fillColor: [ :p | ((p isFragment)&( (p processName = 'aes128DesencryptionResult' ) |
( p processName = 'xacmlPolicyEnforcementPoint') )) ifTrue:
[ Color red ] ifFalse: [ Color lightBlue] ];
width: [:p | p activitiesInvoke * 10];
height: [:p | p activitiesMisc * 10].
view interaction forwarder.
view node: each.
view verticalLineLayout center.
].
view edges: compositions from: #yourself toAll: #appliedFragments.
view treeLayout.
view open
```

Para ejecutar este código se debe abrir un *workspace* en la interface principal de *Moose*, escribir y ejecutar la instrucción relacionada a continuación:

```
(AdoreImporter importFromText: AdoreTest new
taosdWithSecureFragments  ) universe viewCCCmsTree
```

B.2. Código visualización Figura 6.3

Este código corresponde al método **viewWithSecureFragments** de la clase **AdoreUniverse**:

```
viewWithSecureFragments

| view |

view := MOViewRenderer new.

self viewWithSecureFragmentsOn: view.
view open
```

El método **viewWithSecureFragments** a su vez invoca un método denominado **viewWithSecureFragmentsOn** cuyo código es el siguiente:

```
viewWithSecureFragmentsOn: view

view interaction action: #inspect.
view shape rectangle .

view nodes: (processes reject: #isFragment) forEach: [:each |
view interaction forward.
view shape label.
(each processName = 'cms_execRescMission')
ifTrue: [ view shape label fillColor: Color red ]
ifFalse: [ view shape label fillColor: Color green].
view node: each processName.
```

```
view nodes: (each activityNames).
].

view nodes: (processes select: #isFragment) forEach: [:each |

(each processName = 'aes128DesencryptionResult') |
(each processName = 'xacmlPolicyEnforcementPoint')
  ifTrue: [ view shape label size: 10; fillColor: Color red ]
  ifFalse: [ view shape label size: 10; fillColor: Color cyan].

     view node: each processName.
     view nodes: (each activityNames).
     view edges: (each applications collect: #block) from: #yourself
  toAll: #targetedActivityNames.
  ].

view edges: processes from: #yourself toAll: #associatedFragments.
view treeLayout
```

Para ejecutar este código se debe abrir un *workspace* en la interface principal de *Moose*, escribir y ejecutar la instrucción relacionada a continuación:

```
(AdoreImporter importFromText: AdoreTest new taosdWithSecureFragments)
universe viewWithSecureFragments
```

Para mayor información sobre desarrollo bajo *Moose* y *Mondrian*, se recomienda consultar el libro on-line *Pharo by Example*, disponible en http://pharobyexample.org/ .

Bibliografía

[1] F. Agung. Concern-oriented model-driven development framework. In B. Clive, editor, *Software Engineering Conference, Australian*, volume 0, pages 527–535.

[2] R. Anaya, D. Hernández, and J. B. Quintero. Practical experience of the application of aspect-oriented approaches in the development of a thematic portal. *Avances en Sistemas e Informática*, 4(1):109–116, 2007.

[3] V. Atluri and W.-K. Huang. An authorization model for workflows, 1996.

[4] J. Bae, H. Bae, S.-H. Kang, and Y. Kim. Automatic control of workflow processes using eca rules. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):1010–1023, 2004.

[5] O. Barais, J. Klein, B. Baudry, A. Jackson, and S. Clarke. Composing multi-view aspect models. In *Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008)*, pages 43–52, Washington, DC, USA, 2008. IEEE Computer Society.

[6] D. Basin, M. Clavel, J. Doser, and M. Egea. *A Metamodel-Based Approach for Analyzing Security-Design Models*, volume 4735 of *Lecture Notes in Computer Science*, pages 420–435. Springer Berlin / Heidelberg, 2007.

[7] D. Basin, J. Doser, and T. Lodderstedt. Model driven security: From uml models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1):39–91, 2006.

[8] Y. Bendaly Hlaoui and L. Jemni Ben Ayed. *Patterns for Modeling and Composing Workflows from Grid Services*, volume 24 of *Lecture Notes in Business Information Processing*, pages 615–626. Springer Berlin Heidelberg, 2009.

[9] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems, 1999.

[10] E. Bertino, L. D. Martino, F. Paci, and A. C. Squicciarini. *Security for Web Services and Service-Oriented Architectures*. Springer Verlag, 2010.

[11] N. Boucké, D. Weyns, and T. Holvoet. Experiences with theme/uml for architectural design of a multiagent system. In *In Multiagent Systems and Software Architecture, Proceedings of the Special Track at Net.ObjectDays 2006*, pages 87–110, 2006.

[12] C. Braga. Models in software engineering. chapter From Access Control Policies to an Aspect-Based Infrastructure: A Metamodel-Based Approach, pages 243–256. Springer-Verlag, 2009.

[13] C. Braga. A transformation contract to generate aspects from access control policies. *Software and Systems Modeling*, pages 1–15, 2010.

[14] R. Breu, M. Hafner, B. Weber, and A. Novak. *Model Driven Security for Inter-organizational Workflows in e-Government*, volume 3416 of *Lecture Notes in Computer Science*, pages 122–133. Springer Berlin / Heidelberg, 2005.

[15] R. Breu, G. Popp, and M. Alam. Model based development of access policies. *Int. J. Softw. Tools Technol. Transf.*, 9(5):457–470, 2007.

[16] S. Buckl, F. Matthes, and C. M. Schweda. Conceptual models for cross-cutting aspects in enterprise architecture modeling. *Enterprise Distributed Object Computing Conference Workshops, IEEE International*, 0:245–252, 2010.

[17] J. Cardoso. *Business Process Quality Metrics: Log-Based Complexity of Workflow Patterns*, volume 4803 of *Lecture Notes in Computer Science*, pages 427–434. Springer Berlin / Heidelberg, 2007.

[18] J. Cardoso, J. Mendling, G. Neumann, and H. A. Reijers. A discourse on complexity of process models (survey paper). 2008.

[19] A. Carton, C. Driver, A. Jackson, and S. Clarke. Transactions on aspect-oriented software development vi. chapter Model-Driven Theme/UML, pages 238–266. Springer-Verlag, Berlin, Heidelberg, 2009.

[20] P. Caserta and O. Zendra. Visualization of the static aspects of software: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints), 2010.

[21] A. Charfi and M. Mezini. Aspect-oriented web service composition with ao4bpel. *European Conference on Web Services*, pages 168–182, 2004.

[22] A. Charfi, H. MÃ¼ller, and M. Mezini. *Aspect-Oriented Business Process Modeling with AO4BPMN*, volume 6138 of *Lecture Notes in Computer Science*, pages 48–61. Springer Berlin / Heidelberg, 2010.

[23] M.-A. Cibran. *Connecting High-Level Business Rules with Object-Oriented Applications: An approach using Aspect-Oriented Programming and Model-Driven Engineering.* PhD thesis, 2007.

[24] a. Cirit and F. Buzluca. A uml profile for role-based access control, 2009.

[25] M. Clavel, V. da Silva, C. Braga, and M. Egea. *Model-Driven Security in Practice: An Industrial Experience*, volume 5095 of *Lecture Notes in Computer Science*, pages 326–337. Springer Berlin / Heidelberg, 2008.

[26] C. Courbis and A. Finkelstein. Weaving aspects into web service orchestrations. In *IEEE International Conference on Web Services*, pages 219 – 226.

[27] J. Crampton. A reference monitor for workflow systems with constrained task execution, 2005.

[28] Y. Demchenko, L. Gommans, C. Laat, A. Taal, A. Wan, and O. Mulmo. Using workflow for dynamic security context management in grid-based applications. In *GRID '06 Proceedings of the 7th IEEE/ACM International Conference on Grid Computing.*

[29] V. Dhankhar, S. Kaushik, and D. Wijesekera. Securing workflows with xacml, rdf and bpel. In V. Atluri, editor, *Data and Applications Security XXII*, volume 5094 of *Lecture Notes in Computer Science*, pages 330–345. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-70567-3_25.

[30] S. Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[31] E. W. Dijkstra. *On the role of scientific thought.* Selected writings on Computing: A Personal Perspective. New York, NY, USA, springer-verlag new york, inc. edition, 1982.

[32] A. Dury, S. Boroday, A. Petrenko, and V. Lotz. Formal verification of business workflows and role based access control systems. *Emerging Security Information, Systems, and Technologies, The International Conference on*, 0:201–210, 2007.

[33] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control Second Edition.* Artech House, 2007.

[34] A. E. Flores, K. T. Win, and W. Susilo. *Secure Exchange of Electronic Health Records*, pages 1–22. IGI Global, 2011.

[35] D. Gasevic, D. Djuric, and V. Devedzic. *Model Driven Engineering and Ontology Development.* Springer Publishing Company, Incorporated, 2nd edition, 2009.

[36] H. Giese and A. Vilbig. Separation of non-orthogonal concerns in software architecture and design. *Software and Systems Modeling*, 5(2):136–169, 2006.

[37] F. Giraldo, M. Blay-Fornarino, and S. Mosser. Introducing security access control policies into legacy business processes. In *Proceedings of the Fifteenth International Enterprise Distributed Object Computing Conference (EDOC'11), IEEE*, Helsinki, Finland, To appear.

[38] T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec. Flexible and efficient workflow deployment of data-intensive applications on grids with moteur. *Int. J. High Perform. Comput. Appl.*, 22:347–360, August 2008.

[39] T. Glatard, J. Montagnat, X. Pennec, D. Emsellem, and D. Lingrand. Moteur: a data-intensive service-based workflow manager. Technical Report I3S/RR-2006-07-FR, I3S RAINBOW team, Univ. Nice-Sophia-Antipolis, March 2006.

[40] G. Greco, A. Guzzo, G. Manco, and D. Sacca. Mining and reasoning on workflows. *IEEE Trans. on Knowl. and Data Eng.*, 17:519–534, April 2005.

[41] R. Gronmo. Can graph transformation make aspect languages for bpel redundant? *Enterprise Distributed Object Computing Conference, IEEE International*, 0:153–162, 2010.

[42] T. Harris. Migration and security in soa.

[43] A. Hovsepyan, S. Baelen, Y. Berbers, and W. Joosen. *Specifying and Composing Concerns Expressed in Domain-Specific Modeling Languages*, volume 33 of *Lecture Notes in Business Information Processing*, pages 116–135. Springer Berlin Heidelberg, 2009.

[44] X. Jin. *Applying Model Driven Architecture approach to Model Role Based Access Control System*. PhD thesis, 2006.

[45] J. Jurjens. Towards development of secure systems using umlsec, 2001.

[46] J. Kienzle, N. Guelfi, and S. Mustafiz. *Crisis Management Systems: A Case Study for Aspect-Oriented Modeling*, volume 6210 of *Lecture Notes in Computer Science*, pages 1–22. Springer Berlin / Heidelberg, 2010.

[47] M. Koch and K. Pauls. *Engineering Self-protection for Autonomous Systems*, volume 3922 of *Lecture Notes in Computer Science*, pages 33–47. Springer Berlin / Heidelberg, 2006.

[48] B. V. Kumar, P. Narayan, and T. Ng. *Implementing SOA Using Java EE*. The Java Series. Addison-Wesley, 2010.

[49] K. Lee, N. W. Paton, R. Sakellariou, and A. A. A. Fernandes. Utility driven adaptive workflow execution. In *International Symposium on Cluster Computing and the Grid (CCGRID)*, 2009.

[50] K. Lee, R. Sakellariou, N. W. Paton, and A. A. A. Fernandes. Workflow adaptation as an autonomic computing problem. In *2nd Workshop on Workflows in Support of Large-Scale Science*, pages 29–34. ACM Press.

[51] A. Lienhard, A. Kuhn, and O. Greevy. Rapid prototyping of visualizations using mondrian. *Visualizing Software for Understanding and Analysis, International Workshop on*, 0:67–70, 2007.

[52] L. F. Londono. Aproximaciones avanzadas de desarrollo de software: planteamiento de problemas. System and Informatics Department, 2010.

[53] V. Markus. Product line implementation using aspect-oriented and model-driven software development. volume 0, pages 233–242.

[54] S. J. Mellor, A. N. Clark, and T. Futagami. Guest editorsíntroduction: Model-driven development. *IEEE Software*, 20:14–18, 2003.

[55] J. Mendling, M. Strembeck, G. Stermsek, and G. Neumann. An approach to extract rbac models from bpel4ws processes. In *WETICE '04 Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*.

[56] M. Minor, R. Bergmann, S. GÃ¶rg, and K. Walter. Towards case-based adaptation of workflows. In I. Bichindaritz and S. Montani, editors, *Case-Based Reasoning. Research and Development*, volume 6176 of *Lecture Notes in Computer Science*, pages 421–435. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-14274-1_31.

[57] J. Montagnat, B. Isnard, T. Glatard, K. Maheshwari, and M. B. Fornarino. A data-driven workflow language for grids based on array programming principles. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS '09, pages 7:1–7:10, New York, NY, USA, 2009. ACM.

[58] S. Mosser. *Behavioral Compositions in Service-Oriented Architecture*. PhD thesis, 2010.

[59] S. Mosser, A. Bergel, and M. Blay-Fornarino. Visualizing and assessing a compositional approach of business process design. In *Proceedings of the 9th international conference on Software composition*, SC'10, pages 90–105, Berlin, Heidelberg, 2010. Springer-Verlag.

[60] S. Mosser and M. Blay-Fornarino. Taming orchestration design complexity through the adore framework, 2010.

[61] S. Mosser, M. Blay-Fornarino, and R. France. Workflow design using fragment composition - crisis management system design through adore. *T. Aspect-Oriented Software Development VII*, 7:200–233, 2010.

[62] S. Mosser, M. Blay-Fornarino, and M. Riveill. *Web Services Orchestrations Evolution: A Merge Process for Behavioral Evolution*, volume 5292 of *Lecture Notes in Computer Science*, pages 35–49. Springer Berlin / Heidelberg, 2008.

[63] D. Mouheb, C. Talhi, V. Lima, M. Debbabi, L. Wang, and M. Pourzandi. Weaving security aspects into uml 2.0 design models. In *Proceedings of the 13th workshop on Aspect-oriented modeling*, AOM '09, pages 7–12, New York, NY, USA, 2009. ACM.

[64] G. Mussbacher and D. Amyot. Extending the user requirements notation with aspect-oriented concepts. In *Proceedings of the 14th international SDL conference on Design for motes and mobiles*, SDL'09, pages 115–132, Berlin, Heidelberg, 2009. Springer-Verlag.

[65] E. Mytilinaiou, V. Koufi, F. Matamateniou, and G. Vassilacopoulos. *A Context-Aware Authorization Model for Process-Oriented Personal Health Record Systems*, pages 46–65 pp. IGI Global, 2011.

[66] T. Okubo and H. Tanaka. Identifying security aspects in early development stages. In *Third International Conference on Availability, Reliability and Security*, pages 1148–1155.

[67] J. Pavlich-Mariscal, L. Michel, and S. Demurjian. *A Formal Enforcement Framework for Role-Based Access Control Using Aspect-Oriented Programming*, volume 3713 of *Lecture Notes in Computer Science*, pages 537–552. Springer Berlin / Heidelberg, 2005.

[68] K. Rao, M. Rao, K. Devi, D. Kumar, and M. Kumar. Web services security architectures using role-based access control. *(IJCSIT) International Journal of Computer Science and Information Technologies*, 1(5):402–407, 2010.

[69] G. Russello, C. Dong, and N. Dulay. A workflow-based access control framework for e-health applications. In *AINAW '08 Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops.*

[70] P. Sanchez, A. Moreira, L. Fuentes, J. Araujo, and J. Magno. Model-driven development for early aspects. *Information and Software Technology*, 52(3):24, 2010.

[71] C. Sell and T. Springer. Context-sensitive adaptation of workflows. In *Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium*, ESEC/FSE Doctoral Symposium '09, pages 1–4, New York, NY, USA, 2009. ACM.

[72] S. Smanchat, S. Ling, and M. Indrawan. A survey on context-aware workflow adaptations. In *6th International Conference on Advances in Mobile Computing and Multimedia, Workshop session MoMM 2008*, pages 414–417.

[73] D. Stein and S. Hanenberg. Why aspect-oriented software development and model-driven development are not the same - a position paper. *Electronic Notes in Theoretical Computer Science*, 163(1):71–82, 2006.

[74] A. R. Teyseyre and M. R. Campo. An overview of 3d software visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15:87–105, 2009.

[75] W. M. P. van der Aalst, M. Weske, and G. Wirtz. Advanced topics in workflow management: Issues, requirements, and solutions. *J. Integr. Des. Process Sci.*, 7:49–77, August 2003.

[76] I. Vanderfeesten, J. Cardoso, H. A. Reijers, and W. Van Der Aalst. Quality Metrics for Business Process Models. In *Fischer, L. (ed.) BPM and Workflow Handbook 2007*, pages 179–190. Future Strategies, May 2007.

[77] H. Wada, J. Suzuki, and K. Oba. Early aspects for non-functional properties in service oriented business processes, 2008.

[78] L. Wang, Z. Huang, and M. Luo. Supporting dynamic workflow adaptation in a dataflow-constrained workflow net. *New Trends in Information and Service Science, International Conference on*, 0:1000–1005, 2009.

[79] X. Wang, Y. Zhang, H. Shi, and J. Yang. Bpel4rbac: An authorisation specification for ws-bpel. In J. Bailey, D. Maier, K.-D. Schewe, B. Thalheim, and X. Wang, editors, *Web Information Systems Engineering - WISE 2008*, volume 5175 of *Lecture Notes in Computer Science*, pages 381–395. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-85481-4-29.

[80] M. Weske. *Business Process Management Concepts, Languages, Architectures*. Springer, springer edition, 2007.

[81] C. Wolter and A. Schaad. Modeling of task-based authorization constraints in bpmn. In G. Alonso, P. Dadam, and M. Rosemann, editors, *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 64–79. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-75183-0_5.

[82] C. Wolter, A. Schaad, and C. Meinel. Deriving xacml policies from business process models. In *Web Information Systems Engineering - WISE 2007 Workshops*, volume 4832/2007 of *Lecture Notes in Computer Science*. Springer.

[83] L. Zhu and Y. Liu. Model driven development with non-functional aspects. *Aspect-Oriented Requirements Engineering and Architecture Design, ICSE Workshop on*, 0:49–54, 2009.

Las ciencias y las artes llevan en sí mismas
la recompensa de los trabajos y vigilias que se les consagra
ANDRÉS BELLO

Departamento de Informática y Sistemas
Escuela de Ingeniería
Universidad EAFIT
www.eafit.edu.co