Desarrollo de Software Orientado a Aspectos, DSOA 2006
Asociado a XV Jornadas de Ingeniería del Software y Bases de Datos
J. Araújo, J. Hernández, E. Navarro y M. Pinto (Eds)
Sitges (Barcelona), Octubre - 2006

HACIA LA RECONFIGURACIÓN DINÁMICA DE ARQUITECTURAS SOFTWARE ORIENTADAS A ASPECTOS

Cristóbal Costa, Jennifer Pérez, José Ángel Carsí

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
e-mail: {ccosta, jeperez, pcarsi}@dsic.upv.es, web: http://issi.dsic.upv.es

Palabras clave: reconfiguración dinámica, reflexión, CBSD, AOSD

Resumen. *Ante la necesidad de muchos sistemas software de adaptarse a cambios en su entorno sin detener su ejecución, y el hecho de que esta parada puede ser crítica, se propone una solución para incorporar reconfiguración dinámica en arquitecturas software orientadas a aspectos. Concretamente, el trabajo se ha aplicado dentro del enfoque PRISMA, incorporando un nuevo aspecto que proporciona reflexión a nivel de instancias y la posibilidad de modificar dinámicamente la configuración de un modelo arquitectónico PRISMA. Dicha reconfiguración puede ser automática, si es el propio sistema en ejecución el que la desencadena debido a cambios en su entorno, como inducida, si es el arquitecto software el que invoca directamente los servicios de reconfiguración.*

1. INTRODUCCIÓN

En la actualidad, la creciente complejidad de los sistemas software, la importancia de optimizar los recursos disponibles y el ajustarse a los tiempos de entrega de los proyectos, desembocan en la necesidad de utilizar arquitecturas software que permitan gestionar eficientemente la complejidad a distintos niveles de abstracción y faciliten la reutilización de los elementos software que las componen.

Una vez el sistema software es puesto en ejecución, posteriores cambios en los requisitos o en el entorno provocan que dicho sistema deba adaptarse a las nuevas situaciones. Sin embargo, la detención completa de todo el sistema para la incorporación de los nuevos cambios es tan costosa que las organizaciones tienden a evitarlas o a realizarlas en períodos de baja actividad, lo que genera como consecuencia que dichos sistemas se vean rápidamente desfasados y las tareas de mantenimiento sean excesivas. Además, en muchos casos los sistemas muy complejos (como los altamente distribuidos y divididos en varios subsistemas

complejos) son puestos en producción mucho antes de haber sido probados completamente o incluso con algunos subsistemas sin estar finalizados. Estas situaciones tienen la necesidad de adaptarse al nuevo entorno sin necesidad de detener todo el sistema. Por ello, éstas requieren mecanismos que den soporte de la reconfiguración dinámica de la arquitectura software [1]. La reconfiguración dinámica hace referencia a los cambios producidos en tiempo de ejecución que afectan a la topología de una arquitectura software (es decir, cambios que afectan al número de instancias de los elementos arquitectónicos en ejecución y a sus interconexiones).

En este artículo se propone una aproximación para proporcionar capacidades de reconfiguración dinámica al modelo PRISMA, el cual se basa en la integración del Diseño de Software Basado en Componentes (DSBC) y en el Desarrollo de Software Orientado a Aspectos (DSOA). El objetivo es permitir el modelado de las características de reconfiguración evitando en lo posible añadir nuevas construcciones sintácticas que hagan más complejo el meta-modelo PRISMA y con ello dificulten la comprensión del mismo.

La estructura del trabajo es la siguiente: en primer lugar se presenta brevemente el modelo PRISMA. En el apartado 3 se describe la propuesta para modelar las capacidades de reconfiguración. Finalmente, en el apartado 4 se presentan las conclusiones y los trabajos futuros.

2. EL MODELO PRISMA

El modelo PRISMA permite definir arquitecturas de sistemas software complejos, distribuidos y reutilizables. Principalmente se caracteriza por la integración que realiza del DSOA y del DSBC. Esta integración se realiza definiendo las características comunes (distribución, coordinación, seguridad, etc.) (*crosscutting-concerns*) de la arquitectura como aspectos. Un aspecto PRISMA es un *concern* común y compartido por el conjunto de elementos arquitectónicos de la arquitectura. Los aspectos son definidos de forma independiente a los elementos arquitectónicos y existen diferentes tipos de aspectos que varían dependiendo de las características del dominio del sistema software. Los elementos arquitectónicos importan los aspectos mediante una referencia, permitiendo de este modo la reutilización de un mismo aspecto por todos los elementos de la arquitectura que necesiten incorporar sus características y evitando el código mezclado dentro del elemento arquitectónico.

Por lo tanto, un elemento arquitectónico PRISMA se caracteriza por estar formado simétricamente [2] por un conjunto de aspectos de distinto tipo, por las relaciones de sincronización entre estos aspectos (*weavings*), y por uno o más puertos que exportan la funcionalidad ofrecida por el elemento arquitectónico. Los elementos arquitectónicos pueden ser componentes, conectores o sistemas [3].

Los sistemas son componentes complejos que, además de las características anteriores, permiten añadir capas adicionales de abstracción y con ello disminuir la complejidad del sistema, mediante la encapsulación de otros elementos arquitectónicos. Los componentes y conectores a través de sus puertos se interconectan entre sí mediante canales de comunicación denominados *attachments*. Además, existen otros canales denominados *bindings,* los cuales se

definen dentro de un sistema y permiten exportar servicios de los elementos arquitectónicos internos al exterior del sistema (ver Figura 1).

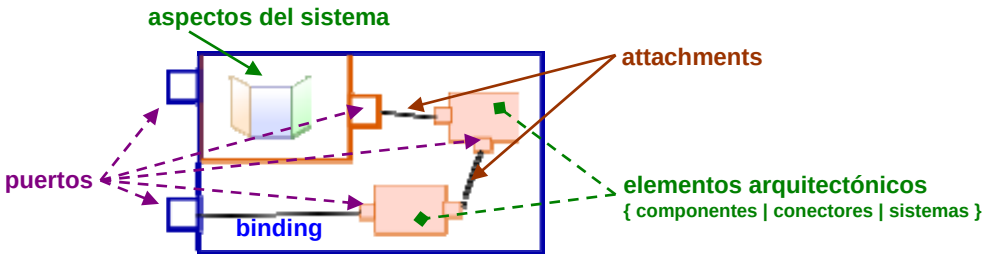

Figura 1. Sistema PRISMA y elementos que lo constituyen

PRISMA divide su lenguaje de descripción de arquitecturas en dos niveles: lenguaje de tipos y lenguaje de configuración [4]. En el primero se definen elementos arquitectónicos y aspectos y en el segundo se definen las instancias y se interconectan entre si con el objetivo de lanzar a ejecución una arquitectura especifica. Dicha arquitectura se conoce por el nombre de configuración inicial.

3. RECONFIGURACIÓN DINÁMICA DE SISTEMAS PRISMA

Un sistema PRISMA encapsula una serie de elementos arquitectónicos interrelacionados entre sí, que se pueden ver como un patrón arquitectónico. La configuración inicial de dichos sistemas, tras su instanciación y ejecución, puede sufrir cambios a lo largo de dicha ejecución. Por lo tanto, debe introducirse un mecanismo que permita modelar y llevar a cabo dicha reconfiguración dinámica evitando introducir artefactos que compliquen el meta-modelo. Además, con el objetivo de proporcionar mayor versatilidad, dicho mecanismo debe permitir realizar la reconfiguración dinámica de dos formas complementarias: (i) de forma *inducida* o *Ad-Hoc*, a través de la invocación directa de los servicios correspondientes o, (ii) de forma *automática* o *programada*, de tal forma que el sistema pueda adaptarse al nuevo entorno cuando detecte determinadas condiciones (por ejemplo, romper una conexión con un elemento arquitectónico al detectar su caída; o instanciar otra copia de un determinado elemento arquitectónico y conectarlo adecuadamente para equilibrar la carga).

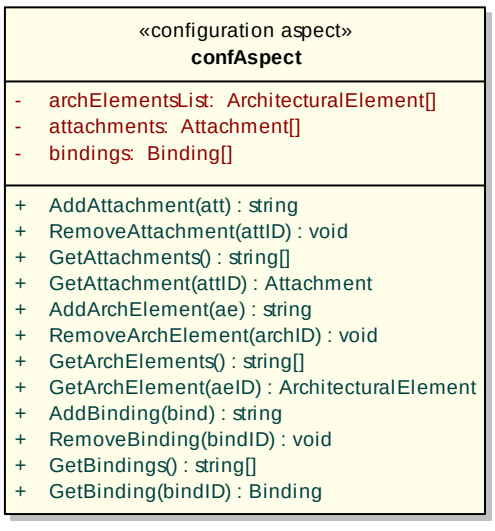

Figura 2. Aspecto de configuración

En este artículo se propone una solución para dar soporte tanto a la reconfiguración dinámica *inducida* como a la *automática* en arquitecturas software orientadas a aspectos. La propuesta consiste en incorporar un nuevo tipo de aspecto (*configuration aspect*, -ver figura 2-) que mantenga actualizado el estado de la configuración en ejecución y que proporcione los servicios necesarios para modificar

dicha configuración.

Dicho aspecto mantiene el estado del sistema al que pertenece mediante tres listas dinámicas: (i) las instancias de los elementos arquitectónicos en ejecución (tanto componentes, conectores como sistemas), (ii) las referencias a los *attachments*¸ que junto a (iii) los *bindings* definen el conjunto de conexiones entre los elementos del sistema. También proporciona los servicios correspondientes para la inserción, eliminación y consulta de dicha información. La implementación de dichos servicios es proporcionada por el *middleware* PRISMANET que permite la ejecución de arquitecturas PRISMA [5], [6], garantizando que la creación y destrucción de *attachments*, *bindings* o elementos arquitectónicos no afecte al resto de elementos en ejecución.

Por ejemplo, la creación de un nuevo *attachment* se llevaría a cabo invocando el servicio *AddAttachment* y pasándole como argumento la instanciación de un nuevo attachment: *new Attachment(puerto1, componente1, puerto2, conector2)*. Del mismo modo, invocando el servicio *GetAttachment* y proporcionando el ID del attachment que se necesita, podría consultarse la información de dicho *attachment*, como son los elementos arquitectónicos que conecta y a través de qué puertos.

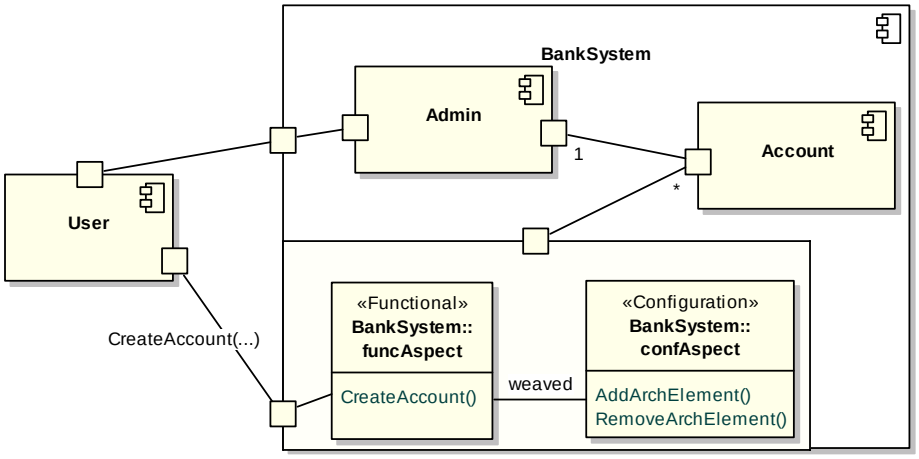

Figura 3. Ejemplo BankSystem

Este aspecto proporciona únicamente la funcionalidad para consultar o modificar la configuración de la instancia en ejecución. La semántica que solicita y consume dichos servicios debe implementarse en cualquiera de los aspectos que constituyen el *sistema*, dependiendo del contexto que origine el cambio. La sincronización entre los distintos servicios de los aspectos será realizada mediante la definición de *weavings* entre ellos.

Por ejemplo (ver Figura 3), dado un sistema PRISMA que modela una entidad bancaria (*bankSystem*) en la cual cada cuenta se modela como un componente (*Account*), la creación de nuevas cuentas sólo se llevará a cabo cuando el servicio *CreateAccount* del aspecto funcional valide que puede realizarse. Una vez este servicio se ejecute con éxito, (1) se disparará el *weaving* relacionado con el servicio *AddArchElement()* del aspecto de configuración y (2) la ejecución de dicho servicio instanciará un nuevo componente de tipo

Account.

Mediante el ejemplo anterior se ha ilustrado el funcionamiento de la reconfiguración *automática*: cada sistema PRISMA posee la capacidad de reconfigurarse a sí mismo y son los aspectos que lo constituyen los que deciden cuándo debe reconfigurarse. En el ejemplo, las nuevas instancias de *Account* sólo serán creadas cuando el sistema determine que es adecuado.

Respecto a la reconfiguración *inducida* o *Ad-Hoc*, una primera aproximación es suponer que cualquier servicio de reconfiguración es accesible directamente por el arquitecto software. Sin embargo, esto viola el principio de encapsulación: componentes de otros proveedores (COTS) no tienen por qué permitir su reconfiguración por terceros. Por otro lado, en sistemas distribuidos no es coherente que el arquitecto software pueda acceder a los servicios de reconfiguración de componentes remotos sino es a través de los canales establecidos por la arquitectura. Es por estas razones que se considera más apropiado limitar el acceso a los servicios de reconfiguración por parte de entidades externas (incluido el arquitecto software), a menos que se hagan accesibles mediante un puerto específico que los exporte. De esta forma, (i) pueden limitarse los servicios de reconfiguración accesibles a un pequeño subconjunto (por ejemplo, permitir la creación de elementos arquitectónicos pero no la modificación de conexiones); (ii) el arquitecto software podrá conectarse remotamente e invocar dichos servicios convirtiéndose en usuario de la arquitectura, es decir, utilizando las conexiones creadas al efecto que enlazan las distintas ubicaciones; y además, (iii) cualquier otro elemento arquitectónico podrá actuar como configurador de otros elementos arquitectónicos, añadiendo expresividad al modelo.

Por lo tanto, la diferencia entre ambas es que la reconfiguración *automática* se produce a consecuencia de la sincronización entre aspectos, a nivel interno y por políticas del propio *sistema*, mientras que la reconfiguración *inducida* es iniciada por entidades externas mediante la invocación de los servicios de reconfiguración exportados por los puertos.

4. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se ha presentado una propuesta para introducir capacidades de reconfiguración dinámica en sistemas basados en componentes mediante la definición de un nuevo tipo de aspecto: el aspecto de configuración. Éste aspecto mantiene la información respecto a la configuración del sistema y permite manipularla en tiempo de ejecución. Al definir toda la semántica de reconfiguración en un aspecto, se evita tener que introducir nuevas estructuras sintácticas en el lenguaje original y por tanto el lenguaje destino (en este caso el AOADL –*Aspect-Oriented Architecture Description Language*– de PRISMA) no se ve afectado, beneficiándose de la nueva funcionalidad.

En el modelo PRISMA, esta aproximación se ha aplicado en aspectos a nivel de *sistema* (componentes compuestos). La misma idea podría aplicarse a un nivel inferior, a nivel de *componentes* (componentes simples), de tal forma que se pudiera consultar y modificar la estructura de un *componente* de forma similar (*puertos*, *aspectos* y *weavings*). Sin embargo, esto plantea problemas adicionales, ya que los cambios en la estructura afectarían a la instancia en ejecución, cuando en realidad deberían afectar a toda la población de

componentes del mismo tipo. Carece de sentido que un *componente* pueda modificar su estructura a nivel de instancia sin propagar dichos cambios al resto de la población de componentes del mismo tipo. Esta cuestión será estudiada con profundidad en trabajos futuros.

Por otro lado, la exportación de los servicios de evolución a través de un puerto con el objetivo de la manipulación *inducida* de la configuración de un determinado sistema, como se ha comentado anteriormente, puede originar serios problemas de seguridad. Cualquier elemento arquitectónico diseñado maliciosamente podría conectarse a dicho puerto y solicitar servicios de configuración que le permitiesen modificar la configuración interna del sistema, cuando esto no debería permitirse. Esto puede resolverse mediante la incorporación de protocolos de seguridad para garantizar qué elementos están capacitados para solicitar determinados servicios. Éste es otro de los trabajos futuros a abordar.

REFERENCIAS

[1] C. E. Cuesta, *Arquitectura de software dinámica basada en reflexión*, Tesis Doctoral, págs. 29-32, ETS Informática, Universidad de Valladolid, Julio 2002.

[2] Harrison W., Harold L., Ossher H., Peri T., *Asymmetrically vs. Symmetrically Organized Paradigms for Software Composition*, IBM Research Report RC22685 (W0212-147). Thomas J. Watson Research Center, IBM, December, 2002.

[3] J. Pérez, N. Ali, J. A. Carsí, I. Ramos, *Dynamic Evolution in Aspect-Oriented Architectural Models*. Second European Workshop on Software Architecture, Lecture Notes in Computer Science, Springer, Vol. 3527. Pisa, June 2005.

[4] J. Pérez, N. Ali, J. A. Carsí, I. Ramos, *Designing Software Architectures with an Aspect-Oriented Architecture Description Language*, 9th International Symposium on Component-Based Software Engineering (CBSE 2006), Lecture Notes in Computer Science, Springer, Vol. 4063. Mälardalen University, Västerås near Stockholm, Sweden, June 29th-1st July 2006.

[5] C. Costa, J. Pérez, N. Ali, J. A. Carsí, I. Ramos, *PRISMANET middleware: Soporte a la Evolución Dinámica de Arquitecturas Software Orientadas a Aspectos*. X Jornadas de Ingeniería del Software y Bases de Datos (JISBD'05). I Congreso Español de Informática (CEDI'05). Granada, septiembre 2005.

[6] J. Pérez, N. Ali, C. Costa, J.A. Carsí, I. Ramos, *Executing Aspect-Oriented Component-Based Software Architectures on .NET Technology*. 3rd International conference on .NET Technologies. Pilsen, Czech Republic, June 2005.