Un modelo de reconfiguración dinámica de arquitecturas

Dennis Neuland[1] and Carlos Canal[2]

[1] Universidad de las Ciencias Informáticas. La Habana. Cuba.
dneuland@uci.cu,
[2] Universidad de Málaga. España.
canal@lcc.uma.es

Resumen Un sistema es reconfigurable cuando permite modificar su comportamiento o estructura sin necesidad de detenerse, y de forma que la reconfiguración sea transparente al usuario en la medida de lo posible. En este trabajo centramos la atención en el desarrollo de un modelo para especificar la reconfiguración dinámica de sistemas. Como aporte exhibe dos especificaciones (una gráfica basada en UML y una basada en notaciones formales) a partir de las cuales pueden ser descritas las arquitecturas de dichos sistemas así como la descomposición del modelo en vistas arquitectónicas. Adicionalmente, se presenta un caso de estudio en el cual es explicada la propuesta.

Palabras claves: reconfiguración dinámica, especificación de arquitecturas, ingeniería guiada por modelos

1. Introducción

Notables esfuerzos en el área de la Ingeniería de Software han estado encaminados a especificar la arquitectura de sistemas distribuidos, en particular aquellos que requieren reconfiguración dinámica de sus componentes. La reconfiguración puede ser necesaria porque los recursos que el sistema está usando dejan de estar disponibles o porque el comportamiento del sistema necesita ser adaptado para remplazar alguno de sus componentes [1].

En este contexto, las operaciones de reconfiguración pueden consistir en la incorporación de nuevos componentes, la actualización de componentes ya existentes o la eliminación de componentes innecesarios. Ante estos cambios, en la nueva configuración pueden producirse problemas de interoperabilidad, pues los nuevos componentes no siempre han sido concebidos para satisfacer todas las necesidades del sistema en el que serán reutilizados.

En este artículo proponemos un modelo para representar arquitecturas de software que permitan la reconfiguración dinámica de sus componentes. El modelo propuesto tiene la particularidad de admitir tanto la especificación gráfica —sustentada en UML— como el estar basado en notaciones formales. Adicionalmente, nuestra propuesta considera el uso de adaptadores para resolver los problemas de interoperabilidad que puedan suceder producto de la reconfiguración.

2. Propuesta de Modelo

Considerando tanto los aspectos estructurales como los de comportamiento necesarios para la interacción y reconfiguración, nuestra propuesta se basa en el desarrollo de un modelo de dos vistas de la arquitectura de un sistema: la vista estructural o estática y la vista de comportamiento o dinámica. Ambas vistas se especifican tanto gráficamente por medio de UML, como formalmente por medio de una notación textual.

Al igual que en [2] nuestra propuesta parte de la concepción de que una arquitectura está compuesta por un número finito de componentes y que una configuración es un subconjunto de estos componentes conectados por medio de contratos de adaptación. Estos contratos de adaptación permitirán abordar problemas de interacción de los componentes en los niveles de signatura y comportamiento.

La vista estática se crea con el objetivo de representar todas las posibles configuraciones en las que puede estar el sistema. Una configuración en esta vista está compuesta por los componentes, sus interfaces (requeridas y proporcionadas) y las relaciones que existen entre ellas.

La vista dinámica está encaminada a modelar las transiciones por las que el sistema puede evolucionar de una configuración a otra, incluyendo las operaciones que deben seguirse en las operaciones de reconfiguración.

Nuestra propuesta extiende [2] e incluye además varios de los aspectos interesantes de otros trabajos relacionados. Entre ellos se pueden mencionar: la consideración de utilizar un componente para administrar la ejecución de las operaciones de reconfiguración (ver [1]) y la especificación del proceso en que pueden ser ejecutadas dichas operaciones (ver [3]).

2.1. UML

Para representar mediante UML la vista estática de una arquitectura se utilizarán elementos como: componentes, interfaces y asociaciones. Para especificar el contrato de adaptación se extenderá el metamodelo en aras de incorporar un nuevo tipo de asociación que posibilite relacionar operaciones de diferentes interfaces. Cada configuración del sistema se concibe como un diagrama de componentes independiente.

En la abstracción de la arquitectura que representa nuestro modelo no se considera necesario especificar el componente que administrará la ejecución de las operaciones de reconfiguración. Su implementación concreta en las arquitecturas específicas puede ser derivada a partir de las transformaciones de este modelo. Sin embargo, sí se considera como un elemento necesario modelar las operaciones de reconfiguración y su comportamiento.

Para tratar con la vista de comportamiento, se propone la utilización de los diagramas de estados de UML. El comportamiento de los componentes se especificará a partir del comportamiento de sus interfaces ofrecidas y requeridas. En ese sentido, a cada una de las interfaces se le asociará un diagrama de estados que representa su comportamiento. En dichos diagramas,

los estados representarán los diferentes modos de funcionamiento de un componente y las transiciones representarán las operaciones ejecutadas por los componentes a través de su interfaces. De la misma forma, se utilizarán los diagramas de estados para especificar el protocolo de los pasos de reconfiguración, en los cuales los estados representan las diferentes configuraciones y las transiciones representan las operaciones de reconfiguración.

2.2. Notación formal

La notación formal de nuestro modelo comienza por la definición de lo que consideramos una arquitectura que soporta la reconfiguración dinámica de sus componentes. Seguidamente, presentamos las definiciones de configuración y sus vistas asociadas.

Definición 1 (Arquitectura). *Una arquitectura SA está definida por un par* $\langle C, R \rangle$*, donde* C *es el conjunto de configuraciones de la arquitectura y* R *es el contrato para aplicar las operaciones de reconfiguración.*

Definición 2 (Configuración). *Una configuración* $c_i \in C$ *se define mediante un par* $\langle csv_i, cbv_i \rangle$*, donde,* csv_i *es la vista estática y* cbv_i *es la vista de comportamiento de la configuración.*

Definición 3 (Vista estática). *Una vista estática csv se define como un par* $\langle P^{sig}, AC \rangle$*, donde,* P^{sig} *es el conjunto de interfaces de signatura de los componentes de la configuración y* AC *es un contrato de adaptación para las interfaces en* P^{sig}*.*

Los contratos de adaptación se especifican por medio de vectores a partir de la propuesta realizada en [4].

Definición 4 (Vista de comportamiento). *Una vista de comportamiento cbv se define como un par* $\langle P^{beh}, S \rangle$*, donde,* P^{beh} *es el conjunto de interfaces de comportamiento de las interfaces en* P^{sig} *y* $\forall\ p_i^{beh} \in P^{beh}\ \exists\ p_i^{sig} \in P^{sig}$*.* S *es el conjunto de estados en* P^{beh}*, que definen los puntos en los cuales puede efectuarse la reconfiguración.*

Definición 5 (Interfaz de comportamiento). *La interfaz de comportamiento* p^{beh} *de un componenente es un LTS, definido mediante una tupla* $\langle S, s_0, L, \rightarrow \rangle$*, donde* S *es el conjunto de estados,* $s_0 \in S$ *es estado inicial.* L *es el conjunto de etiquetas o alfabeto,* $\rightarrow \in S \times L \times S$ *es el conjunto de transiciones.*

Definición 6 (Contrato de reconfiguración). *El contrato de reconfiguración* R *es un LTS, definido mediante una tupla* $\langle CBV, cbv_0, R_{op}, \rightarrow_R \rangle$*, donde* CBV *es el conjunto de configuraciones en su vista de comportamiento y* $cbv_0 \in CBV$ *es la configuración inicial,* R_{op} *es el alfabeto o conjunto de operaciones de reconfiguración y* $\rightarrow_R \subseteq CBV \times R_{op} \times CBV$ *es el conjunto de transiciones (las operaciones de reconfiguración que pueden llegar a suceder en el sistema).*

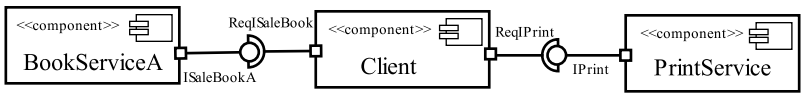
Definición 7 (Operación de reconfiguración). *Una operación de reconfiguración* $r \in \rightarrow_R$ *es una tupla* $\langle s_k^i, r_{op}, s_l^j \rangle$*, donde* $s_k^i \in cbv_i, s_l^j \in cbv_j, cbv_i, cbv_j \in CBV$*, y* $r_{op} \in R_{op}$*.*

Es importante destacar que las operaciones de reconfiguración sólo pueden ejecutarse en determinados estados de la configuración para garantizar la consistencia del sistema. En el caso de una operación de remplazamiento, un estado s_k^i de la configuración fuente cbv_i define en qué momento puede ejecutarse la reconfiguración, mientras que un estado s_l^j de la configuración destino cbv_j define en qué punto de la nueva configuración se reanuda el sistema.

3. Caso de estudio

En esta sección se presenta un caso de estudio de cómo utilizar nuestro modelo. El escenario refleja un cliente que utiliza dos servicios web. El primero de ellos dedicado a la venta de libros y el segundo dedicado a la impresión. En este contexto se prevé que puedan ocurrir varias operaciones de reconfiguración. Por razones de espacio sólo prestaremos atención a las operaciones enfocadas a remplazar el servicio de la venta de libros.

A partir de nuestro enfoque en vistas de la arquitectura comenzamos por modelar la vista estática del sistema usando UML. Dicha vista está compuesta por dos posibles configuraciones. La configuración A (ver figura 3) representa la configuración inicial y la configuración B que se especifica de forma similar y representa la arquitectura tras el remplazo del servicio de venta de libros.



Configuración A

Figura 1. Configuraciones del sistema.

Un ejemplo de contrato de adaptación se muestra en la figura 2. Nótese la presencia de asociaciones que relacionan las operaciones de las interfaces de los componentes.

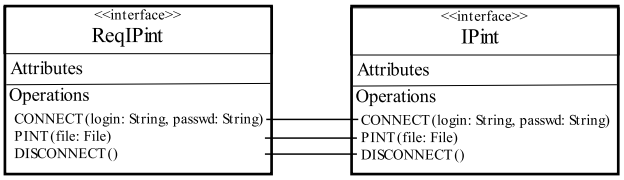


Figura 2. Contrato de adaptación $AC_{Client,Print}$ para Client y PrintService (Configuraciones A y B).

La representación formal de la arquitectura viene dada por:

$SA = \langle \{c_A, c_B, c_C\}, R \rangle$, con
$c_A = \langle csv_A, cbv_A \rangle$, $c_B = \langle csv_B, cbv_B \rangle$, y $c_C = \langle csv_C, cbv_C \rangle$

A continuación se presenta la notación formal de la vista estática de la configuración A. La vista de B puede especificarse de forma semejante:

$csv_A = \langle P^{sig}, AC \rangle$, con
$P^{sig} = \{ReqISaleBook, ReqIPrint, ISaleBookA, IPrint\}$
$AC = \{AC_{Client,BookServiceA}, AC_{Client,Print}\}$

La vista de comportamiento de nuestra arquitectura especificada por medio de UML no será representada por razones de espacio, pero el comportamiento a modelar en los diagramas de estado es parecido al que refleja el LTS de la figura 3. Seguidamente, se presenta la notación formal para esta vista de comportamiento de la arquitectura.

$cbv_A = \langle P_A^{beh}, S^A \rangle$, donde $S^A = \{s_0^A, s_1^A\}$
$s_0^A = \{ReqISaleBook.s_0, ISaleBookA.s_0\}$
$s_1^A = \{ReqISaleBook.s_2, ISaleBookA.s_2\}$

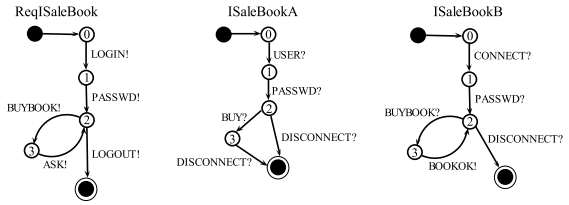


Figura 3. LTS de las interfaces que intervienen en el proceso de compra de libros.

La configuración A presenta dos estados asociados a la reconfiguración del sistema. Ambos corresponden a las operaciones de remplazo del servicio de venta de libros. En el caso particular del servicio de venta de libros, nótese que el mismo puede ser remplazado antes de iniciar una interacción (estado S_0^A) o dentro de una transacción, tras completar las actividades de autenticación (estado S_1^A). Este último escenario garantiza que el cliente pueda continuar con una interacción en este punto aunque el sistema haya remplazado el servicio de venta de libros.

La vista de comportamiento de la configuración B puede ser especificada como:

$cbv_B = \langle P_B^{beh}, S^B \rangle$, donde $S^B = \{s_0^B, s_1^B\}$
$s_0^B = \{ISaleBookB.s_0, ReqISaleBook.s_0\}$
$s_1^B = \{ISaleBookB.s_2, ReqISaleBook.s_2\}$

El contrato de reconfiguración y sus operaciones pueden ser especificadas de la siguiente manera:

$R = \langle \{cbv_A, cbv_B\}, cbv_A, R_{op}, \rightarrow_R \rangle$, con
$\rightarrow_R = \{cbv_A.s_0^A \overset{r_{A,B}}{\rightarrow} cbv_B.s_0^B, cbv_A.s_1^A \overset{r_{A,B}}{\rightarrow} cbv_B.s_1^B, cbv_B.s_0^B \overset{r_{B,A}}{\rightarrow}$
$cbv_A.s_0^A, cbv_B.s_1^B \overset{r_{B,A}}{\rightarrow} cbv_A.s_1^A$, y
$r_{A,B}$ = *replace(BookServiceA,BookServiceB)*,
$r_{B,A}$ = *replace(BookServiceB,BookServiceA)*

4. Conclusiones y Trabajo futuro

Este trabajo presenta una propuesta inicial de modelado de sistemas reconfigurables dinámicamente, basado en trabajos anteriores como [5,2]. La propuesta utiliza UML para representar la arquitectura de los sistemas. Sin embargo, consideramos necesario extender este perfil para representar nuestro modelo de adaptación y reconfiguración. Por otro lado, la propuesta formal que subyace al modelo permite formalizar diversas nociones de reconfiguración y analizar en qué situaciones es posible la reconfiguración de forma que se aseguren diversas propiedades del sistema.

Como trabajo futuro planteamos, tomar como entrada las especificaciones UML para usar un enfoque MDD que permita transformar el modelo UML en una implementación concreta de la arquitectura en plataformas de componentes distribuidos como Fractal o Proactive. Desarrollar un *framework* sobre dichas plataformas que acompañe a las implementaciones y permita la reconfiguración dinámica de acuerdo con el modelo propuesto.

Realizar análisis de propiedades de las operaciones de reconfiguración definidas en el sistema, a partir de la especificación formal del mismo, tal como se propone en [5]. Adicionalmente, se considerarán otras vistas arquitectónicas como la vista semántica. Esta vista facilitará la especificación del significado que tienen las operaciones e interfaces de los componentes en aras de incorporar en nuestro modelo actividades de descubrimiento automático de componentes.

Referencias

1. Almeida, J.P.A., van Sinderen, M., Ferreira Pires, L., Wegdam, M.: Platform-independent dynamic reconfiguration of distributed applications. In: 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, FTDCS'04. (2004) 286–291
2. Canal, C., Cansado, A.: Component reconfiguration in presence of mismatch. Informatica **35** (2011) 29–37
3. Miladi, M.N., Jmaiel, M., Kacem, M.H.: A UML profile and a FUJABA plugin for modelling dynamic software architectures. In: Procs. of the Workshop on Model-Driven Software Evolution (MoDSE'07). (2007) 8
4. Cámara, J., Martín, J.A., Salaün, G., Cubo, J., Ouederni, M., Canal, C., Pimentel, E.: Itaca: An integrated toolbox for the automatic composition and adaptation of web services. In: ICSE, IEEE (2009) 627–630
5. Cansado, A., Canal, C., Salaün, G., Cubo, J.: A formal framework for structural reconfiguration of components under behavioural adaptation. ENTCS **263** (2010) 95–110