Perfil UML para el Modelado de la Integración de Servicios Cloud en Procesos de Desarrollo Incremental

Miguel Zuñiga-Prieto, Silvia Abrahao, Emilio Insfran

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera, s/n, 46022, Valencia, España

{mzuniga, sabrahao, einsfran}@dsic.upv.es

Resumen. En procesos de desarrollo incremental de servicios cloud, la integración de nuevos servicios puede requerir la reconfiguración de la arquitectura actual de la aplicación, siendo importante que dicha reconfiguración sea dinámica para evitar interrupciones en el sistema. En este artículo presentamos un perfil de UML para especificar cómo nuevos servicios deben integrarse en la arquitectura de la aplicación cloud. Esta información de integración es utilizada para generar una nueva orquestación de servicios y los scripts necesarios que actualizan los enlaces entre los nuevos servicios, produciendo por tanto una reconfiguración arquitectónica en tiempo de ejecución. Esta propuesta se ilustra con un caso de estudio práctico en la plataforma Windows Azure© utilizando WCF Workflow para la orquestación de servicios y archivos XML Document Transformation para actualizar la configuración de enlaces de los servicios involucrados.

Palabras Clave: arquitectura de software, reconfiguración dinámica, computación cloud, SoaML, perfil UML.

1 Introducción

Aplicaciones basadas en servicios son sistemas de software compuestos por componentes de software, ya sea producidos en un proceso de desarrollo interno o proporcionados por terceros. Comparadas con aplicaciones empresariales tradicionales, la diferencia más importante reside en su elemento arquitectónico principal, el servicio, el cual encapsula funcionalidades del negocio y actúa como un componente pero con una perspectiva de negocio. Este tipo de aplicaciones generalmente siguen principios de Arquitectura Orientada a Servicios (Service Oriented Architecture – SOA), que promueven la construcción de aplicaciones distribuidas débilmente acopladas utilizando servicios de grano grueso. Adicionalmente, satisfacer principios SOA facilita la interoperabilidad de los servicios que conforman la aplicación; así como la adaptación arquitectónica de la misma, mediante la inclusión, sustitución o eliminación de servicios.

El desarrollo de aplicaciones basadas en servicios usualmente difiere del desarrollo de aplicaciones tradicionales, en donde generalmente: i) los servicios son desarrollados con un enfoque incremental/iterativo; ii) los servicios son componentes reutilizables; y

iii) no necesariamente existe un modelo del sistema con la descripción de todas las funcionalidades a producir [1]. En un enfoque incremental de desarrollo, la integración de los servicios incluidos en un incremento produce cambios (reconfiguración) en la arquitectura actual de la aplicación, siendo importante evitar interrupciones del servicio mientras se produce la integración (reconfiguración dinámica). La reconfiguración dinámica crea y destruye instancias de elementos arquitectónicos en tiempo de ejecución; por lo que, los cambios arquitectónicos que producirá la integración deberían ser planificados y documentados durante la especificación de la integración.

Aplicaciones cloud son aplicaciones basadas en servicios que se ejecutan (o son desplegados) en plataformas cloud provistas por terceros, consumen recursos de la plataforma cloud (ej., entorno de ejecución, colas de mensajes) y siguen un modelo de precios pago por uso. Los servicios cloud que conforman la aplicación generalmente se construyen de acuerdo a la tecnología de la plataforma cloud en la que se desplegaran, lo que conduce a un estrecho acoplamiento entre ellos; requiriéndose por lo tanto mecanismos que faciliten la toma de decisiones y la abstraigan de aspectos tecnológicos.

Creemos que un enfoque de Desarrollo Dirigido por Modelos (Modelm Driven Development - MDD) provee el soporte necesario para la reconfiguración dinámica de arquitecturas de aplicaciones cloud. Un enfoque MDD no solo permitirá documentar los cambios arquitectónicos que producirá la integración de servicios y abstraerlos de aspectos tecnológicos; sino también permitirá reutilizar artefactos de diseño y automatizar el proceso de reconfiguración arquitectónica. El lenguaje para el modelado de servicios SoaML (Service oriented architecture Modeling Language) [2] facilita el modelado y diseño de arquitecturas de servicios mientras sigue un enfoque MDD; sin embargo, en un desarrollo incremental en el que fragmentos de software son desarrollados y entregados en diferentes periodos de tiempo para luego ser integrados, los arquitectos de software deberían ser capaces de describir incrementos utilizando modelos independientes en lugar de forzarlos a trabajar con el modelo general del sistema [3].

Para hacer frente a las necesidades identificadas anteriormente, en este artículo extendemos SoaML, proveyéndolo con características que permiten a arquitectos de software especificar cómo la arquitectura de un incremento de software se integrará en la arquitectura actual del sistema. Seguimos el proceso DIARy (Dynamic Incremental Architectural Reconfiguration) [4], el cual define actividades que soportan la reconfiguración dinámica de arquitecturas de servicios cloud provocada por la integración de nuevos servicios. Siguiendo este proceso, la especificación de la integración es utilizada para generar artefactos cloud (artefactos de software a ser desplegados en plataformas cloud) que ayudan en la reconfiguración de la arquitectura actual.

El resto de este artículo se estructura con las siguientes secciones: la sección 2 describe trabajos relacionados, en la sección 3 se identifican los requisitos que un lenguaje para el modelado de la integración debería satisfacer, en la sección 4 se presenta el perfil UML para el modelado de la integración de servicios, en la sección 5 se aplica el perfil de modelado propuesto, y finalmente en la sección 6 se exponen las conclusiones.

2 Trabajos Relacionados

La reconfiguración dinámica de arquitecturas de software es un área de investigación

abierta, a pesar de que ésta tiene lugar en tiempo de ejecución, la manera en la cual una aplicación es diseñada facilita o dificulta su reconfiguración. En esta sección discutimos como habitualmente se soporta el modelado y la reconfiguración de aplicaciones basadas en servicios desplegados en el cloud.

CloudML [5], es un meta modelo independiente de plataforma cloud para la descripción de los recursos que una determinada aplicación requerirá de plataformas cloud; teniendo en cuenta por el momento únicamente el poner computacional de los recursos (ej., RAM, disco, procesadores). TOSCA [6], es un lenguaje de modelado basado en XML que facilita la especificación de la orquestación y topología de servicios IT. Hace uso de plantillas genéricas para especificar nodos y relaciones entre nodos en una topología de aplicación. Estos trabajos, al igual que otros lenguajes de modelado o marcos de trabajo que incluyen propuestas de modelado para entornos cloud (ej., [5, 6]) soportan el aprovisionamiento y despliegue de servicios cloud; sin embargo, no soportan el diseño o implementación de la arquitectura de estas aplicaciones. En cuanto al soporte para la reconfiguración, se tiene como objetivo facilitar la migración de servicios entre plataformas, más no la reconfiguración de la arquitectura de la aplicación que se produce cuando nuevos servicios son integrados (o migrados).

Con respecto a propuestas de modelado para servicios cloud cuya sintaxis se basa en UML resaltamos dos: CAML y MULTICLAPP. CAML [9] permite representar directamente en UML topologías de despliegue basadas en cloud, y refinarlas con ofertas concretas de proveedores de servicios cloud a través de un perfil UML. MULTICLAPP [10] es un marco de trabajo para el desarrollo de aplicaciones cloud, el cual incluye un perfil UML para modelar aplicaciones cloud como una composición de artefactos de software que pueden ser desplegados en múltiples plataformas cloud. Este marco de trabajo facilita la interoperación entre componentes de diferentes plataformas, y la adaptación (migración entre plataformas) a través de la generación de adaptadores.

De acuerdo a la literatura encontrada, no existen aproximaciones que permitan especificar el impacto arquitectónico que tendrá la integración o migración de nuevos servicios en la arquitectura de una aplicación cloud. Adicionalmente, estas propuestas están orientadas para la migración de servicios cloud y aunque podrían utilizarse en un enfoque de desarrollo incremental, sus lenguajes de modelado no permiten hacer el seguimiento o diferenciar los servicios o componentes que pertenecen a un incremento (o que fueron migrados) de los existentes en la aplicación actual.

3 Requisitos para el Modelado de la Integración

Basado en nuestra experiencia, un lenguaje de modelado que permita especificar cómo la arquitectura de los servicios que componen un incremento se integrará en la arquitectura actual de una aplicación cloud para guiar su reconfiguración debería ofrecer:

1. *Soporte a la especificación parcial de arquitecturas de servicios*: Las aplicaciones basadas en servicio son desarrolladas incrementalmente, por lo tanto fragmentos de una aplicación son desarrollados y entregados en diferentes periodos de tiempo para luego ser integrados. El lenguaje de modelado debería evitar forzar a los arquitectos de software a tener que modificar el modelo de la arquitectura global de la aplicación

actual [3], y por tanto especificar solo los elementos arquitectónicos que describen el incremento a ser integrado. Esto permitirá diferenciar los elementos arquitectónicos del incremento de los de la arquitectura actual, y el posterior seguimiento e identificación de los cambios arquitectónicos producidos por la integración.
2. *Soporte a la especificación del impacto de la integración*: El lenguaje de modelado debería permitir especificar la manera en la cual cada elemento arquitectónico del incremento impacta (o cambia) la arquitectura actual de la aplicación.
3. *Soporte al diseño de servicios desplegados en entornos cloud*: Las decisiones tomadas en las etapas iniciales del proceso de desarrollo (o de un Sprint en un método ágil Scrum [11]) tienen influencia en las decisiones que se tomarán en etapas posteriores. Una de las principales características de los entornos cloud es su capacidad de proveer recursos de acuerdo a la demanda actual. Un lenguaje de modelado debería permitir especificar información acerca de la carga de trabajo que se espera de un servicio, misma que guiará la toma de decisiones en etapas posteriores.

Las descripciones arquitectónicas que utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language - UML) han ganado popularidad y amplia adopción en la industria [3]. Existen varias iniciativas para extender las capacidades de UML con el propósito de modelar arquitecturas orientadas a servicios (ej., [11, 12]); sin embargo SoaML es la más ampliamente adoptada [14]. En este trabajo extendemos tanto UML2.4 como SoaML lo que nos permitirá proveer una solución de modelado para: i) especificar el impacto de la integración utilizando una representación arquitectónica de alto nivel; ii) describir la orquestación de servicios (coordinación de la interacción) separando la lógica funcional de los servicios involucrados de la lógica de la integración, lo que facilitará la reconfiguración arquitectónica.

4 Diseño del Perfil UML

El perfil definido en esta sección (ver Fig. 1), extiende tanto UML2.4 como el perfil de SoaML. Los elementos principales de SoaML que extendemos son:

- *Participante*: Juega el rol de proveedor de servicio, consumidor, o ambos.
- *Arquitectura de Servicios (Services Architecture)*: Define como los participantes trabajan en conjunto proveyendo y usando servicios descritos como *Contratos de Servicio* en una *Arquitectura de Servicios*.
- *Contrato de Servicio (Service Contract)*: Representa un acuerdo entre los participantes involucrados acerca de cómo el servicio será provisto y consumido. Los *Contratos de Servicio* son frecuentemente parte de una o más *Arquitecturas de Servicio*.
- *Uso de Colaboración (Collaboration Use)*: Indica explícitamente como un *Service Contract* o *Services Architecture* es cumplida por las partes que lo componen.

Para evitar que los arquitectos de software modifiquen el modelo de la arquitectura global de la aplicación actual, para modelar los elementos arquitectónicos de la arquitectura del incremento, extendemos la notación de las Colaboraciones UML mediante el estereotipo *ExtendedIncrementArchitecture*. Su semántica es similar a la semántica

de elementos *ServicesArchitecture* de SoaML, pero ésta describe como sus partes internas colaboran para reconfigurar la arquitectura actual.

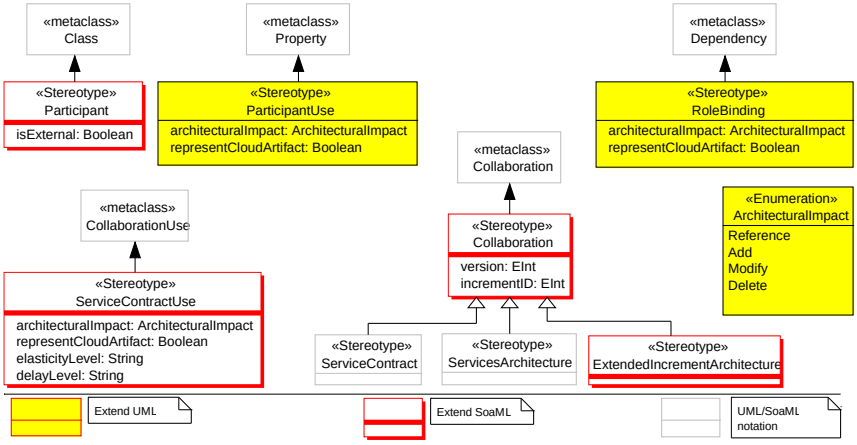

Fig. 1. Perfil de Especificación de Integración

Con el propósito de facilitar le especificación del impacto de la integración de un incremento extendemos las notaciones de los elementos arquitectónicos sujetos a cambios (*ParticipantUse*, *ServiceContractUse* y *RoleBinding*) incluyendo en estas el valor etiquetado *architecturalImpact.* El valor *Add*, *Modify* o *Delete* es utilizado para denotar el impacto que tendrá la integración del elemento del incremento en la arquitectura actual; y *Reference* para denotar que el elemento no cambiará la arquitectura actual de servicios cloud, pero reutilizará uno existente. El valor etiquetado *representCloudArtifact* permite especificar si es que los cambios en elementos arquitectónicos deben ser propagados a los artefactos cloud relacionados durante la implementación.

Debido a que pretendemos dar soporte a la reconfiguración dinámica de arquitecturas de servicios cloud, permitimos que los arquitectos de software describan como se espera que los servicios exploten la gestión de carga de trabajo de los servicios a través del ajuste de su rendimiento. Para hacer frente a cambios en las cargas de trabajo en escenarios elásticos, el uso de Patrones de Gestión de Aplicaciones Cloud [15] sugiere administrar el rendimiento mediante la adición o eliminación de recursos de plataforma cloud; mientras que para escenarios menos elásticos, en donde retardar el procesamiento de las solicitudes podría ser más efectivo que procesarlas inmediatamente, el uso de colas de mensajes es sugerido. Nosotros soportamos ambos escenarios extendiendo el elemento de la notación SoaML *CollaborationUse* con el estereotipo *ServiceContractUse*. Adicionalmente, incluimos los valores etiquetados *elasticityLevel* y *delayLevel*; cuyos posibles valores son: *None*, *Low*, *Medium*, and *High*. Finalmente, las decisiones de implementación pueden variar dependiendo de si un servicio (definido en un *ServiceContract*) es implementado como parte del proceso de desarrollo interno o si el servicio ha sido implementado externamente. Extendemos el elemento *Participant* con el valor etiquetado *isExternal* para indicar si es que se implementarán o no las interfaces correspondientes al rol que el participante tiene en un *ServiceContractUse*.

El perfil UML definido en este trabajo es utilizado para especificar la integración de servicios y forma parte de un proceso de reconfiguración dinámica de servicios (DIARy) que se introduce brevemente a continuación.

5 Aplicación del Perfil UML Siguiendo el Proceso DIARy

En la Fig. 2 se muestra el proceso DIARy (Dynamic Incremental Architectural Reconfiguration) [4], que empieza por modelar una especificación de alto nivel de cómo la arquitectura de un incremento será integrada en la arquitectura de servicios cloud actual. En una segunda actividad la compatibilidad entre las arquitecturas a integrarse es validada. Finalmente, a través de transformación de modelos, la especificación es refinada a una descripción de alto nivel de cómo los artefactos cloud que implementan la arquitectura y hacen uso de servicios de plataforma deberían ser organizados en proyectos; luego transformada a artefactos cloud que implementan elementos arquitectónicos y acciones de reconfiguración arquitectónica, de acuerdo a la plataforma cloud en la que los servicios serán desplegados, entre ellos: i) orquestación de servicios, esqueletos de implementación de interfaces y ii) scripts que cambian la configuración de enlaces (service bindings) de los servicios ofrecidos por los participantes involucrados en la interacción. La reconfiguración arquitectónica es realizada reemplazando en tiempo de ejecución la orquestación de los servicios participantes, la cual es ofrecida como otro servicio; y enlaces entre el servicio de orquestación y los servicios que participan en la interacción.

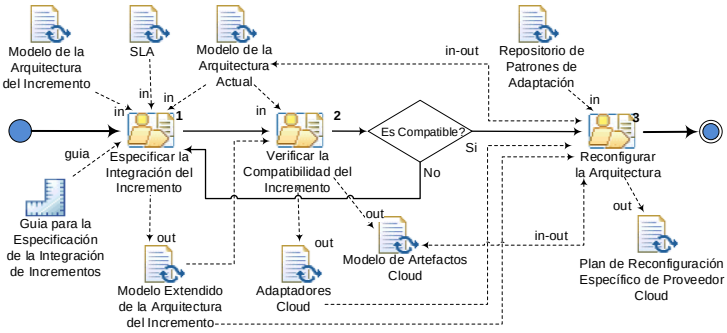

Fig. 2. El Proceso DIARy

Caso de estudio. Para ilustrar la aplicación de nuestro perfil de integración, introducimos a continuación un extracto del caso de estudio de una compañía de manufactura desea mejorar el soporte tecnológico ofrecido a sus socios. Con este propósito, ha iniciado el desarrollo de una aplicación que incorporará funcionalidades de manera incremental. La versión inicial de la aplicación provee servicios cloud a sus distribuidores para que gestionen sus órdenes, facilitando la interacción directa entre el sistema IT de los distribuidores y el de la compañía. Ahora, la compañía requiere incorporar un proceso de entregas como parte de su gestión de pedidos; por lo tanto, en el Incremento-1 proveerá de servicios cloud a sus socios de empresas de reparto, esto les permitirá gestionar órdenes de entrega emitidos por la compañía al transportista.

La siguiente sección describe como el perfil de integración propuesto es utilizado para soportar actividad *Especificar la Integración del Incremento* del proceso DIARy; así como la generación de artefactos cloud que facilitan la reconfiguración dinámica de la arquitectura, paso que es parte de la actividad *Reconfigurar Arquitectura*.

5.1 Soporte a la Actividad *Especificar la Integración de Incrementos*

Las tareas de modelado de esta actividad fueron realizadas utilizando Papyrus, un ambiente de modelado basado en EMF y diseñado como un componente de código abierto en Eclipse. Para describir las entradas de esta actividad, el *Modelo de la Arquitectura del Incremento* (Fig. 3a) y el *Modelo de la Arquitectura Actual* (Fig. 3b) se utilizó el perfil SoaML y el Perfil de Especificación de Integración respectivamente. El elemento arquitectónico «*ServicesArchitecture*» del modelo en la Fig. 3a es una vista de alto nivel de la arquitectura del Incremento_1, el cual, al ser un modelo SoaML incluye el modelado de sus componentes internos: definición de interfaces, descripción de las interacciones validas entre los participantes del servicio (orquestación), mensajes, etc.; sin embargo por restricciones de espacio no se incluyen en este documento. Por otro lado, el *Modelo de la Arquitectura Actual* (Fig. 3b) es utilizado para identificar los elementos arquitectónicos de la arquitectura actual que, luego de la integración, cambiarán o interoperarán con elementos arquitectónicos del incremento.

La integración del Incremento-1 reemplazará el servicio *Place Order* de la arquitectura actual (en negrita en la Fig. 3b) con el servicio *Order With Shipping* (en negrita en la Fig. 3a). Para modelar la integración, el perfil propuesto fue aplicado al *Modelo de la Arquitectura del Incremento* (Fig. 3a); luego el elemento arquitectónico «*ServicesArchitecture*» Increment-1 fue etiquetado utilizando el estereotipo «*ExtendedIncrementArchitecture*»; finalmente sus partes internas fueron etiquetas para especificar como colaborarán para cambiar la arquitectura actual (en negrita en Fig. 3c). Por ejemplo, el «*ServiceContractUse*» Order With Shipping y sus «*RoleBinding*» relacionados fueron etiquetados con *architectural Impact = Add*. Adicionalmente, el «*ServiceContractUse*» purchase:PlaceOrder y sus «*RoleBinding*» relacionados, que no son parte del elemento «*ExtendedIncrementArchitecture*» pero serán remplazados durante la integración, fueron incluidos y etiquetados con *architectural Impact = Delete* (en negrita-cursiva en Fig. 3c). Finalmente, se ha previsto que el servicio *Order With Shipping* tendrá periodos de alta demanda y debe ser atendido sin retrasos, por lo tanto fue etiquetado con los atributos *elasticityLevel = High* y *delayLevel = None*.

5.2 Soporte a la Actividad *Reconfigurar Arquitectura*

En esta actividad se generan artefactos cloud que implementa los servicios y la facilita la operacionalización de la reconfiguración arquitectónica. Sin embargo, describiremos únicamente cómo la información de la especificación es utilizada para la generación de artefactos cloud que permiten la reconfiguración dinámica a través de cambios en las configuraciones de enlaces de los servicios. Para cambiar la configuración de la arquitectura los servicios (nuevos y actualizados) deben estar implementados y desplegados. Para la implementación de la aplicación del caso de estudio utilizamos la plataforma

Microsoft Azure© por ser la más adecuada para aplicaciones basadas en SOA [15]. Los servicios fueron implementados como servicios web WFC (Windows Communication Foundation) utilizando el entorno de desarrollo Visual Studio 2013 (VS). Los servicios ofrecidos por los participantes fueron creados como servicios Web Role mientras que el servicio de orquestación fue creado como servicios WCF Workflow.

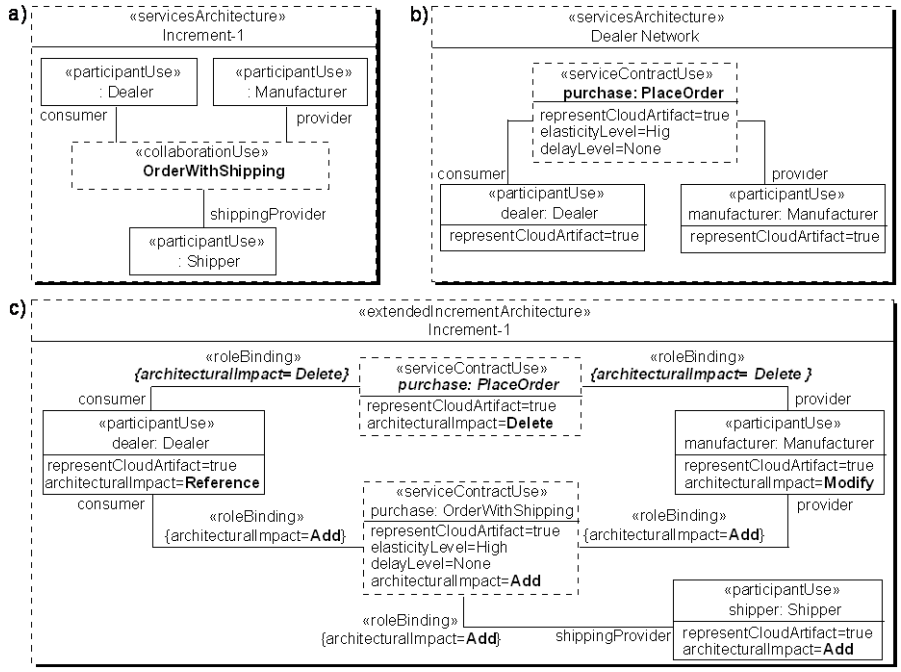

Fig. 3. (a) Modelo de la Arquitectura del Incremento, (b) Modelo de la Arquitectura Actual, (c) Modelo Extendido de la Arquitectura del Incremento

Los archivos de configuración en Microsoft Azure© tienen representación XML y pueden ser modificados aplicando archivos de transformación XML Document Transformation (XDT)[1] durante el despliegue de servicios mediante VS; sin embargo, en lugar de utilizar VS, empleamos el motor de trasformación XDT Transformation Tool[2] para aplicar las transformaciones a los archivos de configuración ya que obtenemos mayor flexibilidad en la gestión de transformaciones. Los archivos XDT incluyen nodos que serán utilizados para cambiar archivos de configuración (.XLS); en donde, atributos *xdt:Transform* indican el tipo de cambio (ej., Insert, Replace, Remove) y atributos *xdt:Locator* definen el criterio de búsqueda de los nodos que serán modificados**.**

Diseñamos transformaciones de modelo para generaran archivos XDT; en donde, reglas de transformación establecen relaciones entre valores *xdt:Transform* y valores del atributo *architecturalImpact* establecidos en la especificación del incremento para

[1] https://msdn.microsoft.com/en-us/library/dd465326(v=vs.110).aspx
[2] https://ctt.codeplex.com/

elementos arquitectónicos «*RoleBinding*». El servicio provisto por el participante *Dealer*, quien inicia la interacción, no fue afectado por la integración del Incremento-1; sin embargo, este servicio inicia la interacción mediante la invocación al servicio de orquestación que será reemplazado; por lo tanto su configuración debe ser actualizada. La información de enlace que utiliza para establecer comunicación con el servicio de orquestación (ver Fig. 4a en negrita) fue actualizada con el endpoint expuesto por el servicio de orquestación *Order With Shipping* (ver Fig. 4b en negrita).

a)
```
<ServiceConfiguration serviceName="Dealer"
 osFamily="4" osVersion="*" schemaVersion="2014-06.2.4">

 <Role name="DealerServices">
  <Instances count="2" />
  <ConfigurationSettings>
   <Setting name="Microsoft.WindowsAzure.Plugins…"
    value="DefaultEndpointsProtocol=https;AccountName=….." />
   <Setting name="testingDelay" value="1000" />

   <Setting name="EndPointPurchase"
    value="http://../PlaceOrder.xamlx" />

   <Setting name="BindingPurchase" value="basicHttpBinding" />
   <Setting name="BindingConfigurationPurchase" value="BasicHttp..." />
  </ConfigurationSettings>
 </Role>
</ServiceConfiguration>
```

b)
```
<ServiceConfiguration serviceName="Dealer"

 xmlns:xdt="http://schemas.microsoft.com/
 XML-Document-Transform">

 <Role name="DealerServices">
  <ConfigurationSettings>

   <Setting name="EndPointPurchase"
    value="http://../OrderWithShipping.xamlx"
    xdt:Transform="Replace"
    xdt:Locator="Match(name)" />

  </ConfigurationSettings>
 </Role>
</ServiceConfiguration>
```

Fig. 4. Extractos: (a) Configuración actual, (b) Archivo de transformación de configuración

El despliegue de un nuevo servicio de orquestación (o cuando un servicio involucrado en la interacción expone un nuevo endpoint) requiere actualizaciones en la configuración del servicio de orquestación. Esta se hace de manera similar a la explicada anteriormente pero dependerá de los valores *architecturalImpact* de los elementos «*RoleBinding*» relacionados al «*ServiceContractUse*» al que pertenece la orquestación.

6 Conclusiones

En un proceso de desarrollo incremental, la integración de nuevos servicios puede producir la reconfiguración dinámica de la arquitectura actual de la aplicación, cambiando su estructura y comportamiento en tiempo de ejecución. En este artículo presentamos un perfil que extiende SoaML para proveerlo de características que permitan a arquitectos de software especificar cómo nuevos servicios deben integrarse en la arquitectura actual de una aplicación cloud. La especificación es utilizada para, a través de transformación de modelos, generar artefactos de software que implementan: orquestación de servicios y actualizaciones a las configuraciones de enlaces de los servicios que interoperarán como resultado de la integración. Estos artefactos son utilizados durante la integración para cambiar la arquitectura de la aplicación sin interrumpir su ejecución.

El proceso DIARy define actividades que soportan la reconfiguración dinámica de arquitecturas de servicios cloud y ha sido utilizado como proceso metodológico para aplicar perfil de integración propuesto. Se ha presentado un caso de estudio práctico en la plataforma Windows Azure© utilizando WCF Workflow para la orquestación de servicios y archivos XML Document Transformation para actualizar la configuración de enlaces de los servicios involucrados.

Actualmente estamos trabajando en el refinamiento del proceso metodológico y en

el desarrollo de casos de estudio más complejos. Se tiene previsto desarrollar servicios dedicados que brinden soporte a la reconfiguración, cuya responsabilidad será gestionar el estado y las relaciones de las instancias en ejecución correspondientes a los servicios afectados por la reconfiguración. Además se proveerá de un soporte de herramienta que integre el conjunto de tecnologías utilizadas y por último diseñar experimentos que permitan validar esta aproximación con usuarios estudiantes y de la industria.

Agradecimientos. Investigación soportada por el proyecto Value@Cloud (MICINN TIN2013-46300-R); Facultad de Ingeniería de la Universidad de Cuenca, Ecuador y Microsoft Azure Research Awards.

Referencias

1. S. Lane, Q. Gu, P. Lago, and I. Richardson, "A Pragmatic Approach for Anal. and Design of Service Inventories," in Service Oriented Computing and Applicat., 2013, pp. 1–19.
2. A. J. Berre, "Service Oriented Architecture Modeling Language (SoaML)-Specification for the UML Profile and Metamodel for Services (UPMS)," Object Manag. Gr., 2008.
3. I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What Ind. Needs from Architectural Languages," IEEE Trans. Softw. Eng., vol. 39, no. 6, pp. 869–891, 2013.
4. M. Zuñiga-Prieto, J. Gonzalez-Huerta, S. Abrahao, and E. Insfran, "Towards a Model-Driven Dynamic Architecture Reconfiguration Process for Cloud Services Integration," in MODELS: Models and Evolutions, 2014, pp. 52–61.
5. E. Brandtzæg, S. Mosser, and P. Mohagheghi, "Towards CloudML, a Model-based Approach to Provision Resources in the Clouds," in 8th European Conference on Modelling Foundations and Applications (ECMFA), 2012, pp. 18–27.
6. T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "TOSCA: Portable Automated Deployment and Management of Cloud Applications," in Advanced Web Services, 2014.
7. C. Chapman, W. Emmerich, F. G. Márquez, S. Clayman, and A. Galis, "Software Architecture Definition for on-demand Cloud Provisioning," Cluster Comput., vol. 15, 2012.
8. F. Leymann, C. Fehling, R. Mietzner, A. Nowak, and S. Dustdar, Moving Applications To the Cloud: An Approach Based on Application Model Enrichment, vol. 20, no. 3. 2011.
9. A. Bergmayr, J. Troya, P. Neubauer, M. Wimmer, and G. Kappel, "UML-based Cloud Application Modeling with Libraries, Profiles, and Templates," in In Proc. Workshop on CloudMDE, 2014, pp. 56–65.
10. J. Guillén, J. Miranda, J. M. Murillo, and C. Canal, "A UML Profile for Modeling Multicloud Applicat.," in Service-Oriented and Cloud Computing, 2013, pp. 180–187.
11. K. Schwaber, Agile Project Management with Scrum. Microsoft Press, 2004.
12. S. Johnston, "UML 2.0 Profile for Software Services," IBM Dev. http//www. ibm. com/developerworks/rational/library/05/419_soa, 2005.
13. V. Ermagan and I. Krüger, "A UML2 Profile for Service Modeling," Model Driven Eng. Lang. Syst., vol. 4735, pp. 360–374, 2007.
14. I. Todoran, Z. Hussain, and N. Gromov, "SOA Integration Modeling: An Evaluation of how SoaML Completes UML Modeling," IEEE 15th Int .Enterprise Distrib. Object Comput. Conf. Work., pp. 57–66, Aug. 2011.
15. A. K. Muppalla, N. Pramod, and K. Srinivasa, "Efficient Practices and Frameworks for Cloud-Based Application Development," in Software Engineering Frameworks for the Cloud Computing Paradigm, Springer, 2013, pp. 305–329.