Razonamiento Basado en Casos para la Materialización de Arquitecturas Orientadas a Servicios

Guillermo Rodríguez, Ezequiel Scott, Álvaro Soria y Marcelo Campo

ISISTAN Research Institute (CONICET-UNICEN) Campus Universitario, Paraje Arroyo Seco, B7001BBO, Tandil, Bs. As.

{guillermo.rodriguez, ezequiel.scott, alvaro.soria, marcelo.campo}@isistan.unicen.edu.ar

Resumen La Arquitectura Orientada a Servicios (SOA) se ha convertido en un paradigma dominante para el desarrollo software distribuido. La mayoría de las organizaciones explotan SOA mediante el descubrimiento y la reutilización de servicios accesible a través de Internet. Sin embargo, determinadas organizaciones necesitan el control interno de las aplicaciones e implementarlas con determinados atributos de calidad. La implementación de una SOA teniendo en cuenta atributos de calidad (por ejemplo, performance, interoperabilidad, seguridad, etc.) requiere que los diseñadores exploren soluciones alternativas; esto resulta una tarea que consume mucho tiempo y es propensa a errores, incluso para diseñadores expertos. La elección de una alternativa de implementación que no tiene en cuenta los principales atributos de calidad del sistema puede dar lugar a desajustes entre el comportamiento prescrito por la arquitectura y el exhibido durante su ejecución real, con un impacto negativo en el software. Esta elección es realizada a menudo por expertos que generalmente buscan soluciones ya existentes que han sido adecuadas en contextos similares. Por este motivo, este trabajo propone un enfoque de Razonamiento Basado en Casos (CBR) para asistir a los diseñadores en la exploración de alternativas de diseño de una SOA, que utiliza el conocimiento sobre los atributos de calidad como las principales guías de materialización hacia diseños detallados orientados a objetos. Los resultados preliminares sobre un caso de estudio demuestran la viabilidad del enfoque propuesto.

Palabras Claves: Arquitecturas Orientas a Servicios; Razonamiento Basado en Casos; Atributos de Calidad; Materialización de Arquitecturas.

1. Introducción

Hoy en día, el uso de Arquitecturas Orientadas a Servicios (SOA) ha ganado gran popularidad como estilo arquitectónico para el desarrollo de software, el cual debe ser a la vez de alta calidad y adaptable a los cambios del mercado [6]. Una de las razones por las cuales SOA se usa extensamente es su capacidad para crear rápidamente aplicaciones ensamblando servicios disponibles en Internet, permitiendo a las empresas de software acelerar el desarrollo de aplicaciones y, consecuentemente, el tiempo de su comercialización. Sin embargo, este enfoque

resulta inadecuado para determinadas organizaciones que demandan que sus servicios de software tengan alta prioridad y determinados atributos de calidad, tales como seguridad, flexibilidad, confidencialidad e integridad de los datos [9], ya sea porque el desarrollo de la funcionalidad principal puede ser crítico, por incertidumbre o cambios en el entorno.

Aunque mucho foco se ha puesto en facilitar el descubrimiento de servicios [8], y su reuso en las aplicaciones SOA [4], poco esfuerzo se ha realizado para asistir a los diseñadores en el desarrollo de servicios teniendo en cuenta objetivos de negocio y preservación de atributos de calidad. Algunas líneas de investigación han contribuido en el desarrollo de servicios dentro de las organizaciones mediante la exploración de diferentes técnicas de Inteligencia Artificial para lograr implementaciones de SOA [1,5]. Sin embargo, el abordaje de la problemática sobre la preservación de los atributos de calidad desde el diseño arquitectónico hacia su implementación ha sido relegado, lo que a menudo conduce a desajustes entre el comportamiento prescrito por la SOA y el comportamiento resultante después de su implementación.

En este contexto, es necesario llevar a cabo más investigaciones sobre el desarrollo de los servicios dirigido por los atributos de calidad dentro de una organización de software. El desarrollo de un servicio normalmente se inicia con algún tipo de descripción arquitectónica (por ejemplo, interfaz pública, principales características que deben proporcionarse, entorno operativo, etc.) y un conjunto de atributos de calidad, que son considerados por el diseñador para producir un modelo más concreto del diseño del servicio. Luego, la implementación de una conexión entre servicios se puede lograr, por ejemplo, mediante la aplicación del patrón *Asynchronous Query* o el patrón *Serializer*, dependiendo de los atributos de calidad definidos por la aplicación SOA. Esta decisión sobre el diseño candidato usualmente involucra *trade-off* entre los atributos de calidad requeridos, resultando en una tarea propensa a errores y que consume tiempo [12]. En efecto, los diseñadores necesitan ser asistidos en la elección entre dos o más alternativas de diseño que pueden ser igual de eficaz desde un punto de vista funcional, pero que pueden divergir de la SOA en términos de atributos de calidad. Esta perspectiva de atributos de calidad para implementar una SOA mediante herramientas semi-automatizadas es un campo de investigación poco explorado. Este proceso de refinamiento se denomina *materialización de SOA* [2].

En este artículo, se presenta un enfoque basado en Razonamiento Basado en Casos (CBR) semi-automatizado que permite a los diseñadores explorar alternativas de diseño orientado a objetos para una SOA dada durante el proceso de su materialización. Siguiendo CBR, el problema de diseño es representado por medio de una especificación arquitectónica de la SOA (atributos de calidad, escenarios de calidad y conectores arquitectónicos), y la solución por medio de un diseño orientado a objetos UML[1] que materializa la arquitectura SOA en base a patrones de diseño comúnmente usados en la implementación de Servicios Web. Con el objetivo de evaluar el enfoque, se presenta un caso de estudio donde se

[1] Unified Model Language http://www.uml.org/

evalúa la efectividad de las soluciones orientadas a objetos sugeridas por medio de métricas tales como *precision*, *recall* y *accuracy.* Los resultados mostraron que el enfoque propuesto es un método eficaz para asistir a los diseñadores en el proceso de materialización de SOA, en términos de generación de soluciones que preservan atributos de calidad involucrados en la especificación arquitectónica de dicha SOA. La principal contribución de este trabajo es el desarrollo de una base de casos que almacena el conocimiento acerca de los conectores arquitectónicos, atributos de calidad y patrones de diseño de SOA para el uso práctico. Por otra parte, se enriquece la base de casos con mapeos que vinculan SOA y los patrones de diseño orientado a objetos teniendo como objetivo los atributos de calidad.

El resto del artículo está organizado de la siguiente manera. La sección 2 describe el enfoque basado en CBR, junto con un caso de estudio. En la sección 3 se discuten las lecciones aprendidas del experimento. Finalmente, la sección 4 concluye el artículo e identifica las futuras líneas de trabajo.

2. Enfoque basado en CBR

El enfoque propuesto apunta a la materialización dirigida por conectores arquitectónicos, atributos de calidad y escenarios de calidad en el contexto de SOA, explorando la metáfora de CBR. La mayoría de los diseñadores utilizan experiencias pasadas para reutilizar soluciones de diseño que resuelvan el problema de la materialización de una arquitectura hacia su implementación; por eso, CBR naturalmente se ajusta en este proceso de diseño basado en la utilización del conocimiento de la organización [13]. CBR es una técnica de Inteligencia Artificial que se basa en el uso de experiencias pasadas, reusándolas y adaptándolas para resolver problemas similares a aquellos que se han resuelto anteriormente [10].

La Figura 1 describe nuestro enfoque, en el cual el diseñador da como entrada un problema que consiste del modelado y descripción de una SOA en términos de atributos de calidad, escenarios de calidad y conectores arquitectónicos. Con la descripción del problema, se procede a **recuperar** de la base de casos aquellos casos que hayan sido útiles para resolver problemas similares al problema dado. De esta manera, la relevancia de un caso viene dada por el grado de similitud entre el problema de un caso almacenado y el problema de entrada. En nuestro enfoque un caso Ci se define como una 2-upla $<Pi, Si>$, donde Pi está compuesto por propiedades de un conector arquitectónico, atributos de calidad y escenarios de calidad para ese conector; mientras que Si es la materialización de ese conector en términos de diseños orientados a objetos.

Entonces, el problema Pi es expresando como un vector en el espacio donde cada componente representa cada elemento del problema definido por el diseñador. Estas componentes son los atributos de calidad y su importancia (por ej.: *performance*=0.8, *seguridad*=0.6); el conector a materializar y sus propiedades según la taxonomía de [11] (por ej.: *propiedad=comunicación, tipo=procedure-call, localidad=global*, etc.); y los escenarios de calidad con sus partes (por ej.: *estimulo*=100-usuarios, *fuente del estímulo*=interna, *ambiente*= tiempo-ejecución,

artefacto= sistema, *respuesta*=notificaciones, medida de respuesta=2seg). La solución Si es expresada como un conjunto de hechos y reglas *prolog* que representan la materialización del conector (por ej.: {*class(A), class(B), association(A,B),* ...}). Luego, se utiliza la similitud del coseno entre vectores (ecuación 1) para determinar la relevancia de un caso, donde $e_{\overrightarrow{P_i},k}$representa el valor de la k-ésima componente del i-ésimo problema P. En la ecuación 2 se definen los posibles valores que puede tomar cada componente, dependiendo si la componente es un valor nominal o lineal (discreto o continuo); $\delta_{P_{ik},P_{jk}}$es la δ de Kronecker.

Luego de la evaluación, se selecciona el caso con mayor valor de similitud como caso candidato y es **revisado** por el diseñador para verificar que efectivamente resuelve el problema propuesto. Si la solución es aprobada, la materialización resultante para el conector arquitectónico es la solución almacenada en el caso candidato y el nuevo caso completo es **guardado** en la base de casos para ser reusado en futuras soluciones de problemas. En caso que la solución sea desaprobada, se procede a recuperar el próximo caso relevante.

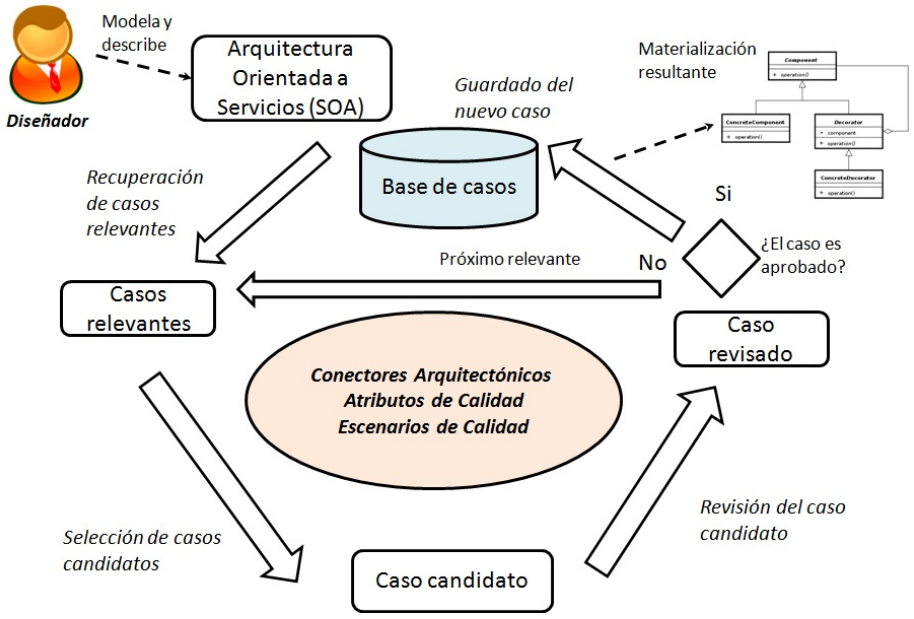

Figura 1. Enfoque basado en CBR.

$$cosineSimilarity(\overrightarrow{P_i},\overrightarrow{P_j}) = \frac{\overrightarrow{P_i}.\overrightarrow{P_j}}{\|\overrightarrow{P_i}\| * \|\overrightarrow{P_j}\|} = \frac{\sum_{k=1}^{K} e_{\overrightarrow{P_i},k} \cdot e_{\overrightarrow{Pj},k}}{\sqrt{\sum_{k=1}^{K} e^2_{\overrightarrow{Pi},k}} \cdot \sqrt{\sum_{k=1}^{K} e^2_{\overrightarrow{Pj},k}}} \quad (1)$$

$$e_{\overrightarrow{Pi},k} = \begin{cases} \left\| e_{\overrightarrow{Pi},k} \right\| & si\, e_{\overrightarrow{Pi},k}\, es\, lineal \\ \delta_{P_{ik},P_{jk}} & otro\, caso \end{cases} \quad (2)$$

2.1. Caso de Estudio

Esta sub-sección describe un caso de estudio que consiste en una aplicación SOA para manejar jugadores de fútbol. La Figura 2 muestra una vista de componentes y conectores de la arquitectura de la aplicación [3]. El sistema permite a los usuarios consultar información relacionada a los jugadores, como por ejemplo, nombre completo, club al que pertenece, posición dentro de la cancha, nacionalidad, etc; asimismo, es posible agregar nuevos jugadores a la base de jugadores y obtener un ranking de jugadores por la cantidad de goles que tienen en su haber. Estas funcionalidades son provistas por el componente cuyo rol es consumir servicios ("SoccerApp Client"). Por otro lado, el componente "SoccerInfo Provider" es responsable de implementar y manejar la forma de recuperar los datos requeridos, como así también, la forma de agregar nuevos jugadores. Estas consultas son realizadas a través del acceso a servicios externos provistos por World Cup Football Pool[2]. Para el consumo y publicación de servicios, la aplicación se conecta con el registro UDDI (Universal Description Discovery and Integration).

Dada la descripción de la arquitectura, el siguiente paso es materializar cada conector. Los conectores especificados por el diseñador son *Service Registry* y *Remote Procedure Call*, los cuales representan los 2 problemas a resolver; mientras que las soluciones almacenadas en la base de casos son patrones de diseño del catálogo de Tomas Erl [7], los cuales han sido empleados con éxito en situaciones anteriores: *Interceptor*, *Delegator*, *Asynchronous Query*, *Producer-Consumer*, *Serializer*, *Abstract-Server*, *Factory Method* y *Service Locator*. Como se ve en el Cuadro 1, si los atributos de calidad para el conector 1 son alta performance y baja interoperabilidad, el enfoque arrojaría como mejor solución el patrón *Asynchronous Query* como se ve en la Figura 2. En cambio, si los requerimientos de calidad son alta interoperabilidad y baja performance, el enfoque devuelve como mejor solución el patrón *Serializer*. Para materializar el conector 2 (conexión directa entre componentes en base a *Remote Procedure Call*), la Figura 2 muestra que el enfoque arrojaría como mejor solución el patrón *Producer-Consumer* si prevaleciera una alta performance por sobre seguridad. En cambio, una solución basada en el patrón *Service Locator* sería el caso mejor relevado por el enfoque si el atributo de calidad a priorizar es seguridad por sobre performance. El Cuadro 1 muestra para cada conector las soluciones sugeridas, junto con la similitud y los atributos de calidad que cada solución intenta maximizar. Para el primer conector, *Asynchronous Query* fue la solución más similar para el problema nuevo (similitud=0.83); mientras que para el segundo conector fue *Producer-Consumer* (similitud=0.998). Ambas soluciones fueron revisadas y aprobadas por un dis-

[2] World Cup Football Pool - http://footballpool.dataaccess.eu/

eñador experto, y materializaron a los conectores descritos. Finalmente, los 2 nuevos casos fueron almacenados en la base de casos.

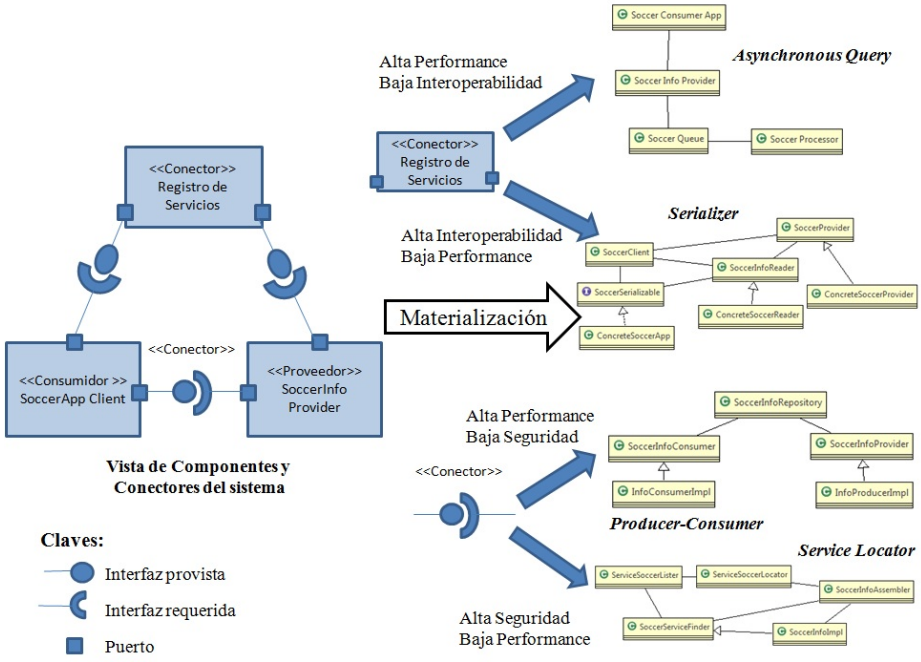

Figura 2. Ejemplo de materialización guiada por atributos de calidad.

3. Lecciones aprendidas

Para realizar el experimento se especificaron diferentes niveles de atributos de calidad, tales como performance, interoperabilidad y seguridad, y observamos diferentes soluciones producidas por la herramienta (Cuadro 1). El Cuadro 1 también resume los resultados en términos de las métricas utilizadas en el experimento. Si bien la precisión promedio es aceptable (71 %), al enfoque aún le faltan casos de entrenamiento y, sobre todo, capacidad para desambiguar situaciones en las que los niveles de atributos de calidad especificados en la arquitectura sean ligeramente diferente. Esto último mejoraría el *Accuracy,* que en promedio es del 45 %. Cabe destacar que el enfoque explora varias alternativas candidatas para la misma arquitectura y es el diseñador quien decide cuál es la alternativa más apropiada para materializar un determinado conector arquitectónico.

Conector	Similitud de soluciones	Solución OO	Trade-offs entre Atributos de Calidad	Precision	Recall	Accuracy
Service Registry	**0.83**	**Asynchronous Query**	Performance e Interoperabilidad	0.67	1	0.4
	0.79	Delegator				
	0.74	Abstract-Server				
	0.72	Serializer				
	0.69	Factory Method				
Remote Procedure Call	**0.998**	**Producer-Consumer**	Performance y Seguridad	0.75	1	0.5
	0.92	Interceptor				
	0.83	Abstract-Server				
	0.82	Delegator				
	0.73	Factory Method				
	0.7	Service Locator				

Cuadro 1. Resultados de la materialización del caso de estudio.

4. Conclusiones

En este trabajo se presentó un enfoque semi-automatizado basado en CBR que asiste a los diseñadores en la exploración de alternativas de materialización preservando atributos de calidad. Las experiencias de materialización (casos en CBR) están estructuradas en términos de problema y solución: el problema está representado por un conector arquitectónico SOA aumentado con información contextual y requerimientos de atributos de calidad, mientras que la solución está representada por una alternativa de materialización en términos de patrones de diseño orientados a objetos. El enfoque facilita la implementación de una SOA manteniendo su integridad conceptual; sin embargo, no genera una aplicación en ejecución, sino que proporciona un modelo orientado a objetos, a partir del cual los diseñadores pueden derivar en la implementación de la SOA.

Una limitación de este enfoque es que los diseños orientados a objetos generados no están mapeados a ninguna plataforma SOA comúnmente usada; en efecto, como trabajo futuro se intentará realizar este mapeo para mejorar las guías de implementación para los desarrolladores. Otra línea de trabajo futuro es modelar los aspectos de comportamiento de los diseños SOA, de modo que este enfoque pueda complementar las soluciones orientadas a objetos con diagramas de secuencia. Finalmente, la mejora en la fase de adaptación de casos es otra línea a seguir, de modo que los nuevos casos almacenados sean lo más cercanos posible a la solución esperada por los diseñadores.

Agradecimientos

Agradecemos el soporte financiero de este trabajo provisto por CONICET y ANPCyT a través del PICT 2011-0080.

Referencias

1. M Agni Catur Bhakti and Azween B Abdullah. Autonomic computing approach in service oriented architecture. In *Computers & Informatics (ISCI), 2011 IEEE Symposium on*, pages 231–236. IEEE, 2011.
2. Marcelo Campo, Andrés Díaz Pace, and Mario Zito. Developing object-oriented enterprise quality frameworks using proto-frameworks. *Software: Practice and Experience*, 32(8):837–843, 2002.
3. Paul Clements, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers, and Reed Little. *Documenting software architectures: views and beyond*. Pearson Education, 2002.
4. Marco Crasso, Alejandro Zunino, and Marcelo Campo. A survey of approaches to web service discovery in service-oriented architectures. *Journal of Database Management (JDM)*, 22(1):102–132, 2011.
5. Islam Elgedawy. Automatic generation for web services conversations adapters. In *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, pages 616–621. IEEE, 2009.
6. John Erickson and Keng Siau. Web services, service-oriented computing, and service-oriented architecture: Separating hype from reality. *Journal of Database Management (JDM)*, 19(3):42–54, 2008.
7. Thomas Erl. *SOA design patterns*. Pearson Education, 2008.
8. J Caleb Goodwin and David J Russomanno. Ontology integration within a service-oriented architecture for expert system applications using sensor networks. *Expert Systems*, 26(5):409–432, 2009.
9. Tibor Kremic, Oya Icmeli Tukel, and Walter O Rom. Outsourcing decision support: a survey of benefits, risks, and decision factors. *Supply Chain Management: an international journal*, 11(6):467–482, 2006.
10. Ramon Lopez De Mantaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, Michael T Cox, Kenneth Forbus, et al. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(03):215–240, 2005.
11. Nikunj R. Mehta, Nenad Medvidovic, and Sandeep Phadke. Towards a taxonomy of software connectors. In *Proceedings of the 22nd international conference on Software engineering*, ICSE '00, pages 178–187, New York, NY, USA, 2000. ACM.
12. Bedir Tekinerdoğan and Mehmet Akşit. *Synthesis-based software architecture design*. Springer, 2002.
13. German L Vazquez, J Andres Díaz-Pace, and Marcelo R Campo. A case-based reasoning approach to derive object-oriented models from software architectures. *Expert Systems*, 27(4):267–290, 2010.