Resumen de Conclusiones [*]

Pedro J. Clemente [**] Diego Sevilla [***] Rafael Corchuelo [****]

El taller sobre *Nuevas Tecnologías de la Información: Componentes y Servicios Web*, es un referente científico de primer orden en cuanto al estudio, implantación y perspectivas de las nuevas tecnologías de la información. En la edición de 2003 se debatieron temas tan interesantes actualmente para el mundo de la Ingeniería del Software como los siguientes:

- Componentes y tecnología grid, nuevas plataformas como OGSA;
- Interoperabilidad de componentes entre distintas plataformas;
- Calidad de servicio en componentes software;
- Búsqueda y recuperación de componentes en repositorios, búsquedas semánticas;
- Configuración de componentes;
- Adaptabilidad de sistemas software basados en componentes;
- Comparativa entre plataformas de componentes;
- Servicios web y su aplicabilidad en sistemas empresariales.

Con el objetivo de dar mayor dinamismo al debate, éste fue dirigido hacia un conjunto de preguntas sobre los temas de interés previamente identificados. A continuación presentamos las reflexiones obtenidas agrupadas por sesiones temáticas.

1. Sesión sobre Interoperabilidad de Sistemas

1.1. ¿Es posible adaptar componentes de distintas plataformas?

La adaptación de componentes de distintas plataformas es un tema que aún está abierto, debido principalmente al creciente impulso que tienen las nuevas tecnologías, las cuales evolucionan tan rápidamente que no dan lugar a la elaboración de estándares de interconexión. En este sentido, la incorporación de elementos intermedios, como la utilización de XML, facilita la interoperabilidad entre sistemas software basados en plataformas heterogéneas. Sin embargo, la inclusión de esta nueva capa de comunicación entre sistemas heterogéneos incrementa el tiempo de ejecución ya que los participantes en la comunicación deben empaquetar/desempaquetar los datos de la comunicación en formato texto (XML). Además, se deben plantear otro tipo de problemas como es la seguridad, ejecución de transacciones,

[*]Parcialmente financiado por la CICYT con los proyectos TIC2002-04309-C02 y TIC2000-1106-C02-01.
[**]University of Extremadura, Cáceres (SPAIN), jclemente@unex.es
[***]University of Murcia, Murcia (SPAIN), dsevilla@um.es
[****]University of Sevilla, Sevilla (SPAIN), corchu@lsi.us.es

Característica	J2EE	.NET
Tipo de tecnología	Estándar	Producto
Vendedores	+30	Microsoft
Interprete	JRE	CLR
Páginas web dinámicas	JSP	ASP.NET
Componentes	Middle-Tier EJB	Componentes .NET
Acceso a bases de datos	JDBC SQL/J	ADO.NET
Servicios web: SOAP, WSDL, UDDI	Sí	Sí

Cuadro 1: Características J2EE vs .NET

garantías de calidad de servicio, etc. Por ello, esta interoperabilidad está limitada, aunque sin duda esta es una de las líneas de investigación abiertas actualmente.

La utilización de servicios web como mecanismo para ofrecer servicios a plataformas heterogéneas, es uno de los mayores avances en cuanto a la interoperabilidad de sistemas software, estos contemplan ya los protocolos adecuados para la localización de servicios, ejecución remota de los mismos, empaquetado y desempaquetamiento de los datos.

1.2. ¿Cómo se pueden conectar componentes .NET con J2EE?

La interoperabilidad entre estas dos plataformas está siendo estudiada con detenimiento por distintas fuentes, ya que ambas constituyen los modelos de referencia para el desarrollo de sistemas software basados en componentes, especialmente dirigido al desarrollo aplicaciones que gestionen procesos de negocio. Las empresas se encuentran en una continua evolución, continuos movimientos empresariales (fusiones, divisiones, etc.) así como cambios en la dirección estratégica como apertura de nuevos mercados, evolución de los procesos de negocio para hacerlos más competitivos, adopción de nuevas tecnologías, en definitiva, una globalización de los servicios que se ofrecen y se requieren. En este marco económico, es común que las compañías combinen, intercambien información y tiendan a consolidar sus procesos productivos, adoptando servicios que proveen otras compañías, las cuales disponen de plataformas heterogéneas, como .NET y J2EE (ver tabla 1).

En este marco, la interoperabilidad entre estas plataformas puede realizarse desde distintos puntos de vista: Web Services y J2EE Connector Architecture.

- La utilización de servicios web, tal y como se ha comentado en la sección anterior, supone actualmente uno de los mecanismos más útiles a la hora de conectar plataformas heterogéneas como .NET y J2EE. Sin embargo, se trabaja en mecanismos para describir adecuadamente características transaccionales, seguridad, etc.

- J2EE Connector Architecture, constituye una aproximación para permitir que los servidores de aplicaciones J2EE y EIS (Enterprise Information Servers) mantengan una compatibilidad adecuada, ya que en la actualidad, en numerosas ocasiones la interacción entre componentes J2EE no resulta satisfactoria. J2EE Connector Architecture ofrece un estándar al cual los proveedores de servidores de aplicaciones pueden acogerse, de este modo, cualquier EJB podrá interactuar adecuadamente con su servidor de aplicaciones, permitiendo una mayor interoperabilidad. En cuanto a la interoperabilidad entre J2EE y .NET, JCA ofrece un mecanismo para permitir la interoperación entre EJB de Sun y componentes COM de Microsoft. Actualmente existen varios servidores J2EE con soporte JCA, además,varias empresas están desarrollando conectores

hacia JCA, de este modo, es posible construir conectores que envuelvan componentes COM, permitiendo la interacción con componentes EJB.

2. Sesión sobre Componentes Software

2.1. Seguridad y certificados de los componentes

El tema de la documentación de componentes está en boca de todos, ya que actualmente, la única documentación que se aporta en la mayoría de los casos es de índole sintáctica, sin tener en cuenta la semántica de los componentes, es decir, su comportamiento. Sin embargo, el proceso de comprobación debe realizarse, independientemente que este pueda o no realizarse de forma automática.

En este sentido, si la comprobación es a nivel de arquitectura de software, es el desarrollador (arquitecto) del sistema quien tiene que realizar estas comprobaciones mediante las garantías de las pruebas aisladas de esos componentes antes de incorporarlos (adaptarlos) al sistema. Si la comprobación es dinámica, además de la comprobación de certificado, sería necesario la aplicación de operaciones de emparejamiento para realizar comprobaciones a nivel de los protocolos (para comprobar si el orden de los mensajes de entrada y de salida coinciden) y también a nivel funcional (pre/post condiciones).

En otro sentido, durante el despliegue de los sistemas software basados en componentes se debe garantizar que la interconexión entre los componentes se realiza entre partes previamente autenticadas, las cuales puedan demostrar que son quienes dicen ser, y hacen lo que dicen que hacen. Para ello, la incorporación de documentación adecuada a los componentes, unidos a un protocolo fiable de comunicación inicial entre los mismos debe garantizar la interconexión.

2.2. ¿Qué tienen que aportar a la seguridad y certificación las tecnologías Grid?

La tecnología Grid puede ayudar a dar soporte a un gran número de usuarios y recursos utilizando una infraestructura de seguridad no centralizada e integrada. En concreto, en [4] se propone un cambio de mentalidad, olvidándose de los conceptos autenticación y autorización, y sustituyéndolos por reconocimiento y confianza, provenientes de la tecnología Grid.

Unos de los principales problemas a los que se tienen que enfrentar las tecnologías Grid es el de la integración de recursos heterogéneos. La heterogeneidad se da a todos los niveles, implicando también las distintas políticas de seguridad de distintas empresas, instituciones e incluso países que forman parte del *Grid*. Por ello, la computación Grid se tiene que basar en mecanismos de seguridad psicológicamente aceptables, a la vez que robustos.

Por su parte, los sistemas Grid pueden aportar un banco de pruebas de un sistema de seguridad a gran escala, dando lugar a nuevas técnicas de gestión de certificados, control de la seguridad distribuida, etc.

2.3. ¿Cómo adaptar componentes a nuevos requisitos del sistema con mínimo esfuerzo?

En este apartado la tecnología de componentes puede verse beneficiada por una adecuada separación de conceptos, de forma que el cambio en los requisitos esté bien localizado y, por lo tanto, afecte solamente a determinadas partes del sistema.

La utilización de contenedores que manejan distintos servicios (seguridad, transacciones, gestión de eventos, ciclo de vida de los componentes, etc.) tal y como sucede en las plataformas EJB y CCM, facilitan la configuración de los componentes, y por tanto permiten adaptar el sistema a nuevas necesidades, simplemente reconfigurando estos contenedores. Los contenedores, tienden a ser cada vez más versátiles y a tratar cada vez más facetas de la ejecución de componentes en el sistema software. Sin embargo, pese a ello, los contenedores aparecen con un conjunto limitado de servicios que puede configurar, lo cual impide adaptar el componente a otros requisitos diferentes a los inicialmente descritos en la plataforma.

Por otro lado, la utilización de técnicas basadas en la separación de aspectos (*Aspect Oriented Programming*) facilitan el desarrollo modular de sistemas software basados en componentes, permitiendo posteriormente de forma semi-transparente la adición de nuevos servicios, normalmente no funcionales, que permiten modificar y adaptar el comportamiento de los componentes.

3. Sesión sobre Búsqueda de Componentes

3.1. ¿Debemos restringirnos a búsquedas sintácticas?

En primer lugar, y como en casi todo lo relacionado con el software basado en componentes, dependería del contexto. Si las actividades de búsqueda se reducen a comprobaciones de signaturas o de tipos, las información sintáctica es suficiente. Esto ha sido lo normal en la mayoría de los trabajos de búsqueda de componentes, ya que generalmente han estado ligados con la clase de especificación de un componente software, normalmente con información sintáctica relativa a las interfaces.

Sin embargo, se ha visto que la especificación de un componente no solamente debería contemplar información sintáctica, sino también otro tipo de información adicional útil para las operaciones de compatibilidad o de reemplazabilidad, como la información semántica tanto a nivel de protocolos (conocido también como coreografía) o a nivel de comportamiento (normalmente recogido como pre/post condiciones); o también información de empaquetamiento, o de implantación (deployment), entre otra información. Estas operaciones de compatibilidad o de reemplazabilidad son generalmente usadas en actividades de búsqueda, y por tanto deberían contemplar restricciones de búsquedas más complejas sobre todas las partes de la especificación de un componente, no solamente la sintáctica, como se ha visto.

3.2. ¿Cómo describir las características semánticas de un componente?

La información semántica de un componente puede referirse tanto al comportamiento como a la coreografía. Para el caso de la información semántica a nivel de comportamiento existen diversos formalismos como las pre/post condiciones, Larch, JML (Java Modeling Language), OCL (Object Constraints Language), lenguajes basados en XML para la descripción de recursos (RDF), o lenguajes de programación como Eiffel, SPARK, las ecuaciones algebraicas, el cálculo de refinamiento, OOZE, VDM++, Object-Z, entre otros. Para el caso, de la coreografía existen diferentes propuestas como redes de Petri, máquinas de estados finitas, lógica temporal, π-cálculo, Object Calculus, Piccola o ASDL (Architectural Style Description Language).

Como se puede comprobar, no existe en la bibliografía una notación estándar y casi todas ellas a su vez son ampliamente utilizadas en diferentes ámbitos de la informática. No obstante, esto no debería suponer un problema, ya que, en teoría, la persona que desarrolla

el componente lo hace con base a que éste debe ser reutilizable y por tanto debería ofrecer un operador de emparejamiento para la notación finalmente usada definir su especificación semántica, facilitando así las comprobaciones de compatibilidad del lado del cliente. Este operador podría existir en el mercado, o en su caso, desarrollarlo el propio desarrollador (o a la inversa, desde el lado cliente). Por tanto, podríamos tener un componente donde su semántica de comportamiento se ha especificado en notación OCL y su semántica de coreografía en π-cálculo, junto con dos operadores para hacer emparejamientos en OCL y en π-cálculo. Estos operadores podrían funcionar como servicios web o CGIs tradicionales.

3.3. ¿Cómo buscar el componente que hace falta en mi aplicación?

La solución a este problema pasa por la utilización de un elemento intermedio, de un mediador, de forma que un cliente describe en un documento/contrato las características del componente que busca y el mediador se encargará de acceder a un repositorio de componentes para obtener aquel componente que se ajuste mejor a las necesidades del cliente. Actualmente, existe el modelo de mediación ODP, pero sólo trabaja con CORBA.

Para mejorar estas búsquedas, también se están utilizando técnicas de inteligencia artificial para clasificar los componentes con base a su comportamiento y conseguir que éstas sean más eficientes. Así, nos podemos encontrar con propuestas que hacen una especificación formal del comportamiento del componente y se da una especificación de los requisitos que se buscan. También para una posterior adaptación se ha de especificar formalmente la arquitectura en la que se va a desplegar el componente.

4. Sesión sobre Servicios Web

4.1. ¿Es posible gestionar la lógica de negocio de una organización mediante servicios web?

Actualmente, esta cuestión es motivo de debate en la comunidad científica, encontrándose razonamientos tanto a favor como en contra. Por un lado, la lógica de negocio representa una de las partes más críticas del sistema y los desarrolladores normalmente tienden a usar técnicas y herramientas estándares ampliamente usadas (en tiempo y espacio). Además, por el momento, no parece que sea una buena solución la gestión de una organización mediante servicios web por las limitaciones que presenta hoy día el modelo de especificación WSDL, como la ausencia de semántica (comportamiento y protocolo), falta de modelos de seguridad, transacciones, etc.

Por otro lado, autores como los de [5] proponen la construcción de una capa de lógica de negocios sobre múltiples servicios web. En este artículo se concluye que construyendo una capa lógica sobre múltiples servicios web, se puede elevar la lógica de negocios residente en la aplicación y llevar la tecnología de servicios web a otro nivel.

4.2. ¿Qué ventaja aportan los servicios web y el modelo OGSA al Grid? ¿Son aplicables los servicios web a tecnología Grid, normalmente ligada al cálculo intensivo y a la computación de altas prestaciones?

OGSA (*Open Grid Services Architecture*) [2] es una propuesta para estandarizar la siguiente generación de plataformas para el Grid [1] basándose en el concepto de servicio web. Aunque todavía está en proceso de cambio, OGSA ya proporciona alguna funcionalidad básica para

construir Grids computacionales: introspección, registro de eventos, gestión del ciclo de vida, asignación de nombres y creación de servicios.

Los servicios web aportan principalmente tres ventajas a la computación Grid [3]:

1. En la computación Grid se necesitan mecanismos para registrar y buscar definiciones de interfaces y descripciones de implementaciones, debido a la necesidad de dar soporte de forma dinámica a la búsqueda y composición de servicios en entornos heterogéneos. Los servicios web cubren este requisito porque proporcionan mecanismos estándares para definir las interfaces de manera separada a su implementación.

2. La adopción de los mecanismos de los servicios web implica que un framework basado en servicios web puede sacar partido de las numerosas herramientas y servicios existentes.

3. Normalmente los servicios de computación de altas prestaciones son ofrecidos por grandes centros de computación, que ofrecen sus servicios a través de servicios web.

5. Conclusiones

Tanto el desarrollo de software basado en componentes, como la utilización de servicios web cada día están más extendidas. Ello nos hace intuir un amplio futuro a ambas tecnologías, las cuales deberán aprender a coexistir, y serán utilizadas principalmente en aquellos sectores donde con más ventajas puedan ser explotados. En este sentido, dada la robustez de las plataformas de componentes serán estos los que formen la capa de modelo de negocio, dejando a los servicios web para formar la capa de presentación, los cuales facilitan la interconexión entre distintas plataformas y ofreciendo una interfaz común para el intercambio de información.

Referencias

[1] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[2] I. Foster, C. Kesselman, J.Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. service infrastructure wg, global grid forum. http://www.globus.org/research/papers/ogsa.pdf.

[3] P. Sherad. Grid architecture blueprint builds on web services, February 2002. http://www.internetnews.com/xSP/article.php/973221.

[4] O. Storz, P.V. Boddupalli, N. Davies, A. Friday, and M. Wu. Leveraging the grid to provide a global platform for ubiquitous computing research. Technical Report CSEG/2/03, University of Lancaster, 2003.

[5] R. Zade and A. Moharil. Building a business logic layer over multiple web services: Leveraging multiple web services to build a truly distributed web services architecture. *Web Services Journal*, 3(8), 2003.