Una Comparación de ESBs desde la Perspectiva de la Integración de Aplicaciones *

Rafael Corchuelo[1], Rafael Z. Frantz[2], Jesús González[3]

[1] Universidad de Sevilla, ETSI Informática
Avda. Reina Mercedes, s/n. Sevilla 41012
corchu@us.es
[2] Universidade Regional do Noroeste do Estado do Rio Grande do Sul (Unijuí)
São Francisco, 501. Ijuí 98700-000 RS (Brasil)
rzfrantz@unijui.edu.br
[3] Intelligent Dialogue Systems, S.L. (Indisys)
Edificio CREA, Avda. José Galán Merino, s/n. Sevilla 41015
j.gonzalez@indisys.es

Resumen. Los ecosistemas software actuales combinan aplicaciones desarrolladas sobre las más dispares plataformas. Con frecuencia aparece la necesidad de integrar algunas de esas aplicaciones para que puedan trabajar de forma conjunta y producir valor añadido, lo que supone retos tanto desde el punto de vista conceptual como técnico. Los ESBs son herramientas muy utilizadas en este contexto puesto que ofrecen una respuesta reutilizable a gran parte de estos retos. Por desgracia, son pocos los trabajos que permiten a los ingenieros del software comparar estas herramientas en este contexto particular, además de que suelen tratar tan sólo aspectos de alto nivel, como, por ejemplo, coste de la licencia, calidad de servicio, patrones de integración soportados, servicios de composición y coreografía disponibles, etcétera. Si a esto le añadimos la disparidad de conceptos y vocabularios usados por cada una de ellas, el resultado es que la inversión en tiempo requerida para tomar una decisión puede ser bastante alta. En este artículo ofrecemos un breve resumen del marco de referencia para comparar ESBs que hemos diseñado sobre la base nuestra experiencia con dos proyectos de integración en ecosistemas software reales.

Palabras clave: integración de aplicaciones, ESBs, marco de comparación.

1. Introducción

En las empresas actuales es muy habitual que convivan aplicaciones que han sido adquiridas o desarrolladas conforme dichas empresas han evolucionado y han ido descubriendo nuevos requisitos, dando lugar a ecosistemas software que no siempre son fáciles de gestionar [9]. Un problema frecuente en estos ecosistemas es integrar dos o más aplicaciones de forma que los datos que manejan por separado estén sincronizados o que puedan colaborar para ofrecer nueva funcionalidad o nuevas vistas de datos [7]. Según un reciente informe los gastos de integración superan en una proporción de entre cinco y veinte a los de desarrollo de nueva funcionalidad [13]. No es de extrañar, por lo tanto, la enorme popularidad que las herramientas para construir buses de servicios empresariales (ESBs) están ganando en este contexto, ya que ofrecen la infraestructura necesaria para integrar los sistemas más dispares [3].

En nuestra revisión de la tecnología, hemos estudiado las siguientes herramientas: Camel [5], Mule [4], ServiceMix [2], Spring Integration [11] y BizTalk [16]. Las hemos elegido puesto que

* Financiado parcialmente por el Plan Nacional de I+D+I (expediente TIN2007-64119) y la Órden de Incentivos de la Junta de Andalucía (expediente P07-TIC-02602). Parte de esta financiación procede de fondos FEDER. El trabajo de R.Z. Frantz ha sido financiado por la Evangelischer Entwicklungsdienst e.V. (EED).

actualmente son de las más populares y, además, en el caso de ServiceMix, al estar basada en JBI [2] podemos considerarla como el representante canónico de todos los ESBs que también lo implementan, por ejemplo, OpenESB, Fuse ESB o JBoss ESB. En mayor o menor grado, todas permiten implementar soluciones de integración basadas en el patrón arquitectónico Pipes&Filters [6]. Siguiendo este patrón, una solución de integración se puede ver como el diseño de un conjunto de mensajes que fluyen a través de tuberías entre diversos filtros.

Recientemente, hemos trabajado con estas herramientas para diseñar dos soluciones de integración en un par de ecosistemas software reales, a saber: el sistema de gestión de llamadas telefónicas de Unijuí y un sistema de orientación laboral para una institución pública. Esta experiencia nos ha permitido profundizar en estas herramientas y diseñar un marco de comparación que permite compararlas usando propiedades que hemos clasificado y agrupado en tres categorías distintas que nos permiten establecer un paralelismo con los perfiles profesionales jefe de proyecto, arquitecto software y programador. Los pocos trabajos [15] [10] que hemos encontrado para comparar ESBs, están enfocados casi en su totalidad en propiedades de alto nivel y centradas prácticamente tan sólo en los dos primeros perfiles, como, por ejemplo, las propiedades coste de la licencia, calidad de servicio, patrones de integración soportados, servicios de composición y coreografía disponibles, etcétera.

El resto del artículo está organizado de la siguiente forma: en la sección 2, presentamos dos ecosistemas reales que nos han ayudado a diseñar el marco de comparación que proponemos; en la sección 3 describimos algunas de las propiedades de nuestro marco en relación con el alcance de las soluciones de integración, las capacidades de modelado y aspectos de carácter puramente técnico; finalmente, mostramos nuestras conclusiones más importantes en la sección 4.

2. Escenario motivador

Nuestro interés por desarrollar el marco de comparación que presentamos en este artículo surge a partir del estudio llevado a cabo para diseñar dos soluciones de integración en el contexto de dos ecosistemas reales y complejos completamente diferentes.

El primer ecosistema software explorado fue el de gestión de llamadas telefónicas de la universidad Unijuí, en el que un sistema central de gestión de llamadas denominado Call Center System almacena en una base de datos información sobre todas las llamadas realizadas por los empleados de la universidad desde los teléfonos que ésta pone a su disposición. Cada mes se realiza un análisis de esta base de datos con el objetivo de encontrar aquellas llamadas que suponen un coste para la universidad. El resultado de este análisis es un informe que se remite a los empleados para que marquen en él las llamadas personales. Los empleados deben devolver el informe relleno a los servicios de gestión, que procederán a realizar los cargos oportunos a través de una aplicación denominada Debit System y a notificar el hecho por correo electrónico y SMS. Además, también existe un sistema denominado Personnel System a partir del que se obtienen los números de teléfono, direcciones de correo y otros datos personales de los empleados de la universidad. Nuestro objetivo en este caso ha sido diseñar una solución de integración que permita automatizar el proceso con el objeto de aumentar su eficiencia y reducir los errores que se producen al manejar toda esta información con procedimientos artesanales.

Nuestro segundo caso de estudio está relacionado con el desarrollo de un asistente virtual interactivo para la orientación laboral que la empresa Indisys está realizando en colaboración con otras para una institución pública. El asistente virtual es una aplicación web que responde a preguntas frecuentes del estilo "¿dónde encuentro información sobre ofertas de trabajo?". Este ecosistema cuenta con un sistema llamado Natural Language Understanding que es capaz de reconocer frases como la anterior y convertirlas en una estructura de datos con semántica. Hay otro sistema llamado CORE que se encarga analizar la estructura anterior con el objetivo de determinar si es necesario continuar el diálogo para que el usuario proporcione más información, por ejemplo, "¿cuál es su sector profesional?" o "¿en qué provincia reside?", o por el contrario se tienen todos los datos para interrogar los servidores de información de la institución pública. En este último caso, el problema es que estos

servidores son tremendamente heterogéneos, por lo que otra empresa está desarrollando un sistema para integrarlos y proporcionar una fachada de acceso denominada Knowledge Manager. Sea cual sea el caso, CORE produce una respuesta textual y la envía al sistema Text to Speech, que se encarga de transformarla en voz.

Destacamos tres aspectos fundamentales de estos proyectos:

Aplicaciones: La característica común de todas las aplicaciones integradas en nuestro primer caso de estudio es que constituyen sistemas independientes que fueron desarrollados sin pensar nunca en la posibilidad de que tuvieran que ser integrados con otros. Esto implica que no proporcionan interfaces de programación, por lo que en el caso del Personnel System es necesario acceder directamente a su base de datos, mientras que en el caso del Call Center System, que es un sistema propietario, es necesario interactuar a través de su interfaz de usuario usando un sistema de wrapping [1]. En el segundo caso de estudio, por el contrario, casi todos los sistemas proporcionan una interfaz de programación que facilita la integración, salvo en el caso de los sistemas de información de la institución pública, que son completamente dispares.

Brechas: Las principales brechas a salvar están relacionadas con la tecnología, los modelos de datos y la representación de los mismos. En los dos casos de estudio, hemos encontrado una amplia variedad de tecnologías, desde bases de datos a las que se puede acceder mediante JDBC hasta interfaces de programación que utilizan protocolos propietarios. No es de extrañar, pues, que los modelos de datos usados sean completamente dispares y sean necesarias con frecuencia transformaciones tanto en el esquema como en la representación de los datos.

Restricciones: En todos los casos, las aplicaciones integradas no han sufrido ningún cambio. Esta restricción ha sido motivada en unos casos por el hecho de tratarse de aplicaciones propietarias como el Call Center System; en otros porque el coste de la reingeniería necesario para que ofreciesen una interfaz de programación más adecuada era inasumible. Otra restricción importante ha sido mantener las aplicaciones integradas completamente desincronizadas con el objetivo de que puedan continuar siendo operadas y administradas de forma completamente independiente.

3. Marco de comparación

En esta sección comentamos nueve de las 47 propiedades que forman parte de nuestro marco de comparación, organizadas en tres categorías, a saber: alcance, modelado y técnica. Debido a la limitación de espacio en este artículo, comentamos tan sólo tres propiedades de cada una de las categorías y los valores que creemos que son ideales para cada una de ellas, cf. tabla 1.

3.1. Alcance de las herramientas

La primera categoría de propiedades que hemos identificado está relacionada con el alcance de las herramientas, y suele servir de base para la toma de decisiones de los jefes de proyecto. Se trata de propiedades cuya ausencia puede dificultar enormemente e incluso invalidar una propuesta ya que para suplirlas es necesario implementar extensiones cuyo coste de desarrollo creemos que puede ser inabordable en la mayoría de los casos. Entre las más importantes, destacamos las siguientes:

Contexto: Suele ser habitual distinguir entre los siguientes contextos de integración: Enterprise Application Integration (EAI), en donde el énfasis es integrar aplicaciones con el objetivo de sincronizar sus datos o de implementar nuevas funcionalidades; Enterprise Information Integration (EII), cuyo énfasis están en proporcionar una vista en vivo de los datos que manejan las aplicaciones integradas; Extract, Transform, and Load (ETL), que busca proporcionar vistas materializadas de dichos datos sobre la que aplicar técnicas extracción de conocimiento [14]. En todos los casos anteriores, se asume implícitamente que las aplicaciones integradas forman parte de una misma organización. Recientemente cada vez se le está prestando más atención al

Propiedades	**Camel**	**Mule**	**ServiceMix**	**Spring Int.**	**BizTalk**	**Ideal**
Alcance						
Contexto	EAI	EAI	EAI	EAI	EAI/B2BI	EAI/B2BI/EII/Mashup
Patrón arquitectónico	Filters	Pipes/Filters†	Pipes/Filters†	Filters	Pipes/Filters	Pipes/Filters
Nivel	PSM	PSM	PSM	PSM	PSM	PIM/PSM
Modelado						
Card. filtros	$1:N$	$1:N$	$1:N$	$1:N$	$N:M$	$N:M$
Card. localidades	$N:M$-Comp	$N:M$-Comp	$N:M$-Comp	$N:M$-Comp	$1:1$	$N:M$-Comp/Repl
Correlación	No	No	No	No	Sí	Sí
Técnicas						
Modelo ejecución	$1:1$	$1:1$	$1:1$	$1:1$	$1:1$	$N:M$
Mensajes anómalos	Sí	No	No	Sí	Sí	Sí
Patrón de comunicación	Sí‡	No	Sí	No	No	Sí

† Tan sólo ofrecen soporte parcial para el diseño de filtros.
‡ No permite definir nuevos tipos de MEPs.

Tabla 1. Algunas propiedades del marco de comparación.

problema de integrar aplicaciones pertenecientes a distintas organizaciones con el objeto de implementar procesos de negocio inter-organizacionales; a este contexto se suele hacer referencia como Business to Business Integration (B2BI). También recientemente, han cobrado importancia los denominados mashups, que son aplicaciones que se ejecutan en un navegador web e integran datos proporcionados por diversas aplicaciones web. De nuestro análisis se desprende que casi todas las soluciones estudiadas se encuentran en el contexto de EAI, con la única excepción de BizTalk, que también tiene en cuenta el contexto B2BI.

Patrón arquitectónico: Como ya sabemos, Pipes&Filters es el patrón por excelencia en nuestro campo de interés. Por lo tanto, parece razonable esperar que las herramientas para la construcción de ESBs proporcionen lenguajes específicos de dominio para diseñar tanto tuberías como filtros. Por desgracia, no es así ya que Camel y Spring Integration no proporcionan soporte alguno para el diseño de tuberías y Mule y ServiceMix tan sólo proporcionan un soporte parcial para el diseño de filtros. En el caso de Mule, este soporte se reduce a unas cuantas tareas de transformación o enrutado de mensajes que se deben combinar siempre de forma lineal; en el caso de ServiceMix, la herramienta, como tal, no ofrece ninguna ayuda para construir los filtros, pero existe un componente JBI que proporciona la implementación de algunas tareas comunes.

Nivel de abstracción: Trabajar con modelos independientes de la plataforma (PIM) permite diseñar soluciones estables tan independientes como resulta posible de la tecnología utilizada para implementarlas y su inevitable evolución. Al trabajar con modelos PIM es necesario contar, además, con herramientas capaces de transformarlos en modelos dependientes de la plataforma sobre la que se quiere realizar la implementación (PSM) [8]. Por desgracia, ninguna de las herramientas estudiadas permite realizar una separación clara entre los niveles PIM y PSM.

3.2. Capacidades de modelado

La categoría de modelado representa propiedades que no son tan críticas como las anteriores, y están relacionadas con el trabajo de los arquitectos software. La razón es que en caso de carecer de ellas aún es posible diseñar una solución de integración efectiva con un coste razonable, pero quizás el diseño sea mucho más complejo y menos intuitivo, lo que puede tener, evidentemente, un efecto negativo sobre el mantenimiento posterior. Entre las más importantes, destacamos las siguientes:

Cardinalidad de los filtros: Algunos filtros pueden requerir mensajes de distintas fuentes para poder desempeñar su labor. Un situación común se presenta cuando es preciso tratar las distintas

partes de un mensaje utilizando filtros diferentes y posteriormente combinar los resultados. Por desgracia, tan sólo BizTalk permite diseñar filtros capaces de tomar información de varias fuentes. Por el contrario, todas las propuestas analizadas permiten que un filtro envíe información a varias tuberías de salida simultáneamente. En este punto es interesante destacar que gracias a herramientas como los motores de BPEL es posible diseñar filtros capaces de tomar información de varias fuentes de una forma razonablemente simple, y, por lo tanto, suplir las carencias de las herramientas analizadas, aunque a costa de introducir un nuevo lenguaje en el proceso de modelado.

Cardinalidad de localidades: El término localidad hace referencia a la ubicación física sobre la que se implementa una tubería, por ejemplo, una carpeta en un sistema de archivos. Con la excepción de BizTalk, todas las soluciones estudiadas permiten que varios filtros lean o escriban mensajes desde/en una localidad compartida. Generalmente es interesante distinguir entre dos tipos de lectura: con competencia (Comp), en cuyo caso tan sólo uno de los filtros puede leer en cada momento de una determinada localidad, o con replicación (Repl), en cuyo caso todos los filtros pueden leer al mismo tiempo. Todas las soluciones implementadas permiten lectura con competencia, pero ninguna la lectura con replicación.

Correlación de mensajes: Dado que no podemos asumir sincronía alguna entre las aplicaciones integradas, es muy común que los mensajes lleguen a los filtros de forma desordenada, por lo que es responsabilidad de los mismos correlacionarlos de forma que aquellos mensajes que son complementarios sean tratados siempre de forma conjunta. Esta necesidad es mucho más imperiosa en aquellos casos en que es posible diseñar filtros o tareas con múltiples entradas, por lo que no es de extrañar que tan sólo BizTalk soporte directamente la correlación de mensajes.

3.3. Características técnicas

En esta categoría hemos incluido aquellas propiedades que afectan a la facilidad de programación, al rendimiento o a la gestión de las soluciones de integración, por lo que su ausencia puede dificultar el despliegue y la operación de las mismas. Entre ellas, destacamos las siguientes:

Modelo de ejecución: El modelo de ejecución de los filtros puede tener un impacto importante en el rendimiento de una solución de integración. El más sencillo consiste en asignar una hebra a cada mensaje o conjunto de mensajes que deben ser tratados de forma conjunta por un filtro; por supuesto, las hebras pueden tomarse de un pool para conseguir de esta forma mantener siempre bajo control la carga total de trabajo del servidor. Este es el modelo que implementan todas las herramientas estudiadas, pero presenta una deficiencia que creemos que puede afectar de forma negativa a la escalabilidad de las soluciones. El problema está relacionado con el hecho de que cuando una instancia de un filtro llega a un punto en el que no puede continuar ejecutando tareas, por ejemplo, porque es necesario esperar la llegada de un mensaje, la hebra queda ociosa durante un tiempo completamente indeterminado. De nuestra experiencia concluimos que un modelo capaz de ejecutar de forma asíncrona varias instancias de un mismo filtro sobre un pool de hebras sería mucho más efectivo. En la actualidad estamos trabajando en el diseño e implementación de este modelo con el objetivo de evaluar ambas alternativas y poder obtener conclusiones.

Mensajes anómalos: Cuando un mensaje presenta algún tipo de anomalía que hace imposible que sea procesado por un filtro, lo normal es que éste produzca una excepción y que el mensaje en cuestión se almacene de forma que pueda ser analizado por el administrador del sistema. Además, es muy deseable que estos mensajes también puedan ser tratados de forma automática de manera que se intente llevar a cabo algún tipo de acción correctiva en el mismo instante en el que se detectan. Por desgracia, ni Mule ni ServiceMix ofrecen ningún mecanismo que permita automatizar el tratamiento de estos mensajes.

Patrón de comunicación: El patrón para intercambio de mensajes (MEPs) permite definir tipos específicos de comunicación [12]. Una solución de integración puede utilizar distintos tipos de

MEPs, como el unidireccional y sin respuesta (InOnly), el bidireccional con una respuesta obligatoria (InOut), el bidireccional con una respuesta opcional (InOptionalOut), etcétera. El uso de MEPs facilita la correlación entre los mensajes que llegan a un filtro y las respuestas obtenidas. En nuestro estudio hemos visto que las únicas que soportan este patrón de comunicación son Camel y ServiceMix, aunque sólo ServiceMix permite definir nuevos tipos de MEPs.

4. Conclusiones

La principal conclusión del estudio que hemos llevado a cabo es que ninguna de las herramientas analizadas es ideal, en el sentido de que ninguna de ellas tiene valores óptimos para todas las propiedades examinadas. También hemos detectado algunas carencias importantes en todas ellas. Quizá una de las principales es que ninguna permite desarrollar modelos independientes de la plataforma, lo que liga las soluciones a la tecnología disponible en cada momento y puede dificultar su mantenimiento cuando la tecnología evolucione. Otra carencia importante está en relación con la cardinalidad de filtros, que en la mayoría de los casos tan sólo permiten una fuente de mensajes como entrada; esta limitación ha demostrado en la práctica ser problemática dado que algunas veces el procesamiento de un mensaje exige información extra que se debe buscar en otro recurso integrado, lo que exige otra de entrada de mensajes. Estas conclusiones se derivan de nuestra experiencia práctica en los dos proyectos que nos han servido de motivación para el desarrollo de nuestro trabajo. En breve esperamos poner en marcha una nueva experiencia en colaboración con la empresa Sytia Informática, S.L., en esta ocasión en el contexto de un sistema B2BI para la gestión de imágenes médicas.

Referencias

1. C. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan. Survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, 2006.
2. B.A. Christudas. *Service Oriented Java Business Integration*. Packt Publishing, 2008.
3. J. Davies, D. Schorow, and D. Rieber. *The Definitive Guide to SOA: Enterprise Service Bus*. Apress, 2008.
4. P. Delia and A. Borg. *Mule 2: Official Developer's Guide to ESB and Integration Platform*. Apress, 2008.
5. Apache Foundation. Apache Camel home. Available at http://activemq.apache.org/camel.
6. M. Fowler. *Patterns of Enterprise Application Architecture*. Addison–Wesley, 2002.
7. G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.
8. A. Kleppe, J. Warmer, and W. Bast. *MDA Explained*. Addison–Wesley, 2003.
9. D. Messerschmitt and C. Szyperski. *Software Ecosystem: Understanding an Indispensable Technology and Industry*. MIT Press, 2003.
10. Brenda M. Michelson. Enterprise service bus evaluation framework: Criteria for selecting an enterprise service bus as an integration backbone. Technical report, Patricia Seybold Group, 2005.
11. Inc. SpringSource. Spring integration home. Available at http://www.springframework.org/spring-integration.
12. W3C. Web services message exchange patterns. Available at http://www.w3.org/2002/ws/cg/2/07/meps.html#id2612442.
13. J. Weiss. Aligning relationships: Optimizing the value of strategic outsourcing. Technical report, IBM, 2005.
14. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
15. Robert Woolley. Enterprise service bus (esb) product evaluation comparisons. Technical report, State of Utah - Department of Technology Services, 2006.
16. D. Woolston. *Foundations of BizTalk Server 2006*. Apress, 2007.