Análisis de Calidad en Enterprise Service Bus, propietarios y de código abierto, mediante la utilización del método de calidad ESB-QM

Arias Esteban Gabriel[1], Mg. Patricia Bazán[2]
[1] Facultad de Informática UNLP [2] LINTI Facultad de Informática UNLP
estebangarias@ciudad.com.ar, pbaz@ada.info.unlp.edu.ar

Resumen

La integración de sistemas de manera confiable, escalable, robusta, segura y adaptable a los cambios es una necesidad ante el crecimiento y diversidad de sistemas y tecnología y ante los requerimientos de alta disponibilidad. Como resultado de de estos esfuerzos de integración surge Arquitectura orientada a servicios (SOA por sus siglas en Ingles), que permite construir aplicaciones e integrar las existentes a partir de servicios auto-contenidos y heterogéneos, tanto en tecnología como en locación geográfica. La herramienta tecnológica que surge para resolver la parte técnica de esta arquitectura, que refiere a la integración tecnológica entre servicios, son los ESB (Enterprise Service Bus). En este artículo se propone demarcar los lineamientos para realizar análisis de calidad a atributos que son propios de los productos ESB, basándonos en la taxonomía definida por el método ESB-QM, que es el único enfoque hasta la fecha que los considera.

***Palabras clave*:** ESB, SOA, ESB-QM

Contexto

El presente es un trabajo de fin de carrera de Licenciatura en Informática de la Facultad de Informática de la UNLP, del alumno Esteban Arias, dirigida por la Mg. Patricia Bazán.

Introducción

Hasta hace poco más de una década, las aplicaciones de software empresarial se enfocaban hacia el tipo de información que procesaban y se construían de manera aislada unas de otras [1]. Entre estos sistemas se pueden encontrar:

- ERP (Enterprise resource planning)
- BI (Sistemas de Business Intelligence)
- CRM (Customer Relationship Management)
- Sistemas de Logística
- Sistemas Web (Portales, Aplicaciones Web y Web Services)
- Sistemas pertenecientes a los partners de negocio

Estas aplicaciones están construidas en los más diversos lenguajes y plataformas tecnológicas. Se pueden encontrar por ejemplo: [2]

- Clientes servidores monolíticos, sistemas n-capas y silos mainframe.
- Sistemas orientados a objetos, procedurales y basados en componentes
- Múltiples lenguajes de programación.
- Middleware diferentes para manejar la comunicación (MOMS, Object Request Brokers (CORBA, RMI), RPCs)
- Múltiples modelos de transmisión (conversacionales, requerimiento/respuesta, publicación \ suscripción)
- Múltiples tecnologías de Persistencia de Datos (BD relacionales, VSAM, BD de escritorio como Microsoft Access, XML BD)

En contraste, los procesos de negocio, raramente hacen uso de un silo funcional en particular, mas bien son procesos que usan distinta información procedente de los mismos y a través de distintos departamentos claves de las empresas.

Dicha característica de los procesos de negocio, a la hora de integrar las aplicaciones, generó graves inconvenientes y costes de desarrollo al intentar automatizarlos como consecuencia lógica de las necesidades de procesamiento de las empresas, debido a que la información se encontraba frecuentemente en varios sistemas a la vez, con formatos dispares, y tecnológicamente diferentes, a la vez que podían residir en lugares geográficamente distintos.

a. Evolución hacia ESB

La primera aproximación para integrar las aplicaciones era no tener ningún método preestablecido y realizar las mismas al vuelo, conectando directamente las aplicaciones. A este tipo de integración se la llamo "arquitectura accidental" [3] y se muestra en la figura 1, donde se ve la conexión realizada de manera puntual y directa entre las distintas aplicaciones.

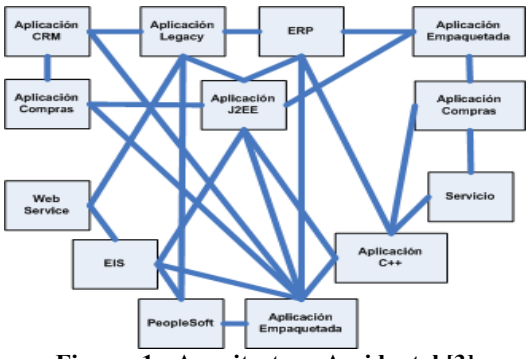

Figura 1 - Arquitectura Accidental [3]

Dicha arquitectura tiene varios inconvenientes como:

- Crecimiento exponencial de conexiones
- Alto acoplamiento entre las aplicaciones
- Altos costo de modificación

- Imposibilidad de unificar monitoreo y gerenciamiento de la infraestructura.
- Imposibilidad de usar estándares

Una mejora al planteo anterior son las arquitecturas Hub and Spoke. Estas se caracterizan por usar un administrador de mensajería centralizado (HUB), construido sobre un middleware orientado a mensajes (MOM), y adaptadores (SPOKE), que conectan las aplicaciones al administrador, como se visualiza en la figura 2, y convierten los datos de las aplicaciones en un formato que el administrador entiende, y viceversa. Como se ve en la tabla 1, introduce mejoras considerables en confiabilidad, acoplamiento, cantidad de conexiones y gerenciamiento de la infraestructura, con respecto a la arquitectura accidental.

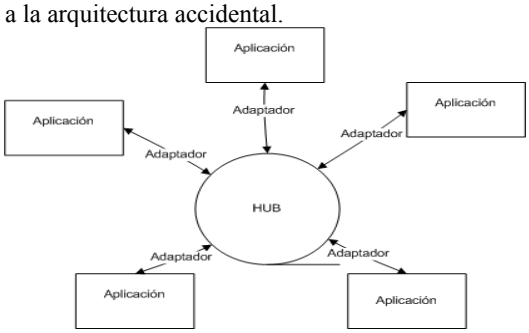

Figura 2 - Arquitectura hub and spoke

Tabla 1 – Aportes de la Arquitectura hub and spoke

Característica mejorada	Descripción de la mejora
Acoplamiento entre aplicaciones	El HUB y los adaptadores aíslan a las aplicaciones de los detalles tecnológicos de ruteo y transformación de mensajes y protocolos, y de las direcciones físicas reales para comunicarse entre ellas. Adicionalmente, permite virtualmente que una aplicación no conozca a las aplicaciones que consumen sus mensajes.
Reducción de conexiones	El orden de Conexiones es O(n), ya que las aplicaciones solamente se conectan al adaptador que les corresponde. El número n es la cantidad de aplicaciones a conectar.
Confiabilidad	Los MOMs existentes en el HUB garantizan el envío de mensajes por medio de técnicas como store and forward.
Gestión centralizada	Al ser una arquitectura centralizada, también su gestión y monitoreo se realiza de manera centralizada y homogénea.

Sin embargo hay algunos problemas aquí:

1. Se introduce un punto único de falla, ya que si falla el HUB fallará toda la infraestructura asociada a él.
2. Costos altos de instalación debido a que no permiten despliegue incremental de la infraestructura, y de licenciamiento y capacitación de la misma, ya que utilizan tecnologías propietarias del proveedor del HUB y sus adaptadores.
3. Baja performance y escalabilidad ante el crecimiento de la información a administrar, por su naturaleza centralizada y monolítica
4. No soportan nuevas tecnologías como Web Services.

b. Enterprise Service Bus

Como resultado de los esfuerzos de resolver los problemas de integración de las arquitecturas anteriores surgió SOA (Service Oriented Architecture).

SOA representa una arquitectura ágil, abierta, federada, extensible y compuesta, comprendida de servicios autónomos con capacidades de calidad (QoS), interoperables, reusables, y con transparencia de locación de los mismos. Permite establecer una abstracción de la lógica y de la tecnología de negocio de manera que se puede introducir cambios al proceso de modelado del negocio y de la arquitectura técnica, dando por resultado bajo acoplamiento entre estos modelos. [1][4].

La arquitectura SOA cumple con integración, reusabilidad, transparencia de locación, interoperabilidad y seguridad, mediante un componente central denominado ESB (Enterprise Service Bus). Un Enterprise Service Bus es una plataforma de integración, basada en estándares y altamente distribuida, que permite a las aplicaciones ser accedida vía servicios. Esta infraestructura combina: [3] [4] [5]:

- Middleware de mensajería (MOMs)
- Web Services y REST-Based Services
- XML
- Transformación de datos
- Conversión de protocolos
- Ruteo inteligente

Una descripción detallada de cómo soporta un ESB estas características se encuentra en la tabla 2.

El modelo de despliegue de los ESB es una red integrada de instancias de servicios, que se distribuyen y colaboran a través de contenedores de servicios [3], como se visualiza en la figura 3. Estos contenedores están distribuidos a través de la red y permiten descentralizar la arquitectura y el despliegue incremental de los mismos.

Un ESB desacopla a los consumidores de los servicios de los proveedores de los mismos, y permite virtualmente, conectar los mismos independientemente de la tecnología de las aplicaciones. Proveen capacidades adicionales como [3] [4]:

- Manejo centralizado de la seguridad
- Envío de mensajes asegurado
- Configuración y monitoreo centralizado
- Directorios de servicios

- Gateways de acceso externo a la empresa
- Servicios de coreografía de procesos.

Los mismos soportan estilos de integración empresarial como son arquitecturas manejadas por mensaje y arquitecturas manejadas por eventos, además de SOA.

Tabla 2 - Aportes fundamentales de ESB

Característica que mejora ESB	Descripción de la mejora
Procesamiento distribuido	EL procesamiento se distribuye a través de contenedores de servicios, que permiten gestionar el despliegue y ciclo de vida de los servicios y puede ser distribuido en distintas locaciones geográficas. Esto permite eliminar el punto único de falla que genera el procesamiento centralizado de las arquitecturas HUB and SPOKE, a la vez que hace escalable fácilmente la infraestructura.
Uso de estándares	A diferencia de las arquitecturas anteriores, hace alto uso de estándares, en especial de XML, lo cual baja costos de capacitación y licenciamiento, permite cambiar de productos ESB con relativa facilidad, y a la vez permite más interoperabilidad entre los distintos componentes que conforman la infraestructura de integración.
Web Services	Los ESB soportan completamente la mediación y gestión de servicios vía web servicies, a la vez que adhieren a la mayoría de los estándares WS-*
Ruteo Inteligente	Un ESB es capaz de decidir el destino de los mensajes que transitan a través de él, basado en el contenido del cuerpo del mensaje, basado en los datos de contexto del mensaje o por medio de motores de reglas como pueden ser los motores que implementan BPEL4WS
Transformación de mensajes, conversión de protocolos y mensajería confiable	Un ESB es capaz de de transformar los datos de un mensaje de un formato a otro, en especial por el uso de plantillas XSLT. También es capaz de mediar transparentemente entre las aplicaciones para realizar conversiones entre protocolos distintos de comunicación. La mensajería confiable se logra a través de los MOMs que posee el ESB.

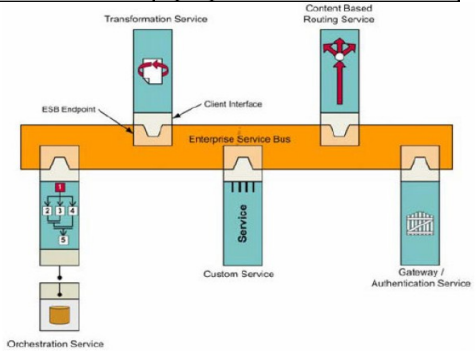

Figura 3 - Visión general de un ESB [6]

c. Métodos de evaluación de ESB

La especificación y evaluación de calidad de productos de software, es un factor clave para garantizar la calidad adecuada de los mismos. Esto puede lograrse mediante la definición de características de calidad apropiadas, teniendo en cuenta los propósitos de uso de los productos de software. Los requisitos no funcionales, son importantes para la integración de aplicaciones, y en consecuencia para realizar comparación entre solucione que las realicen [6].

En particular los modelos de evaluación de calidad de arquitecturas de software reúnen taxonomías de atributos de calidad, comúnmente usados para especificar y evaluar requerimientos no funcionales. Buscan definir características que debe satisfacer un producto de software, de forma que se puedan cuantificar las mismas a través de atributos medibles. La diferencia entre modelos radica en dicha clasificación [6].

Existen varios métodos para evaluar ESB. El grupo Seybold [7], y Vollmer [8] diseñaron frameworks y cuestionarios de evaluación que agrupan las características funcionales y no funcionales que deberían tener un ESB. Estos métodos son bastante completos pero no apuntan a medir los ESB mediante atributos de calidad sino por el grado de resolución de características técnicas. Los atributos de calidad los nombran de manera aislada y mezclados con las características técnicas, como se muestra en la tabla 3 y no tienen en cuenta atributos clave de un ESB como por ejemplo reusabilidad.

Tabla 3 - Algunas características, agrupando capacidades técnicas mezcladas con atributos de calidad [7].

Característica a evaluar	Capacidad
Servicios de integración	protocolos soportados
	Auditoria
Calidad De Servicio	Tolerancia A fallos
	performance y escalabilidad
	control de transacciones
Integracion de la infraestructura empresarial	existencia de herramientas de desarrollo
	existencia de herramientas de testing
	existencia de registros de servicios

Por otro lado los trabajos bien apuntados a medir atributos de calidad de ESB, trabajan con algún atributo de calidad en concreto, como ser performance en [9], y no definen un método completo de evaluación de calidad de ESB.

No se observa a partir del análisis realizado, evaluaciones completas sobre los ESB, direccionadas íntegramente desde el punto de vista de calidad. Una de las aproximaciones evaluadas, que va en la dirección correcta, para brindar una herramienta de evaluación desde este punto de vista de calidad, y que es la que se selección para el presente trabajo es el Modelo ESB-QM.

El modelo de calidad ESB-QM, es una instancia del modelo ISO-9126-2001, clasifica el producto de software usando las 6 características del modelo ISO-9126-1 más cuatro características propias de los ESB, como se visualiza en la tabla 4, la cual compara ambos modelos: ubicuidad, delegabilidad, reusabilidad y manejabilidad

Tabla 4 - Modelo ESB-QM [6]

Característica	Sub-Característica	ISO	ESB-QM
Funcionalidad	Precisión	●	●
	Interoperabilidad	●	●
	Seguridad	●	●
Confiabilidad	Madurez	●	●
	Facilidad de recuperación	●	●
	Tolerancia a Fallos	●	●
	Escalabilidad		●
Usabilidad	Facilidad de aprendizaje	●	●
	Facilidad de entendimiento	●	●
	Facilidad de operación	●	●
Eficiencia	Rendimiento	●	●
	Utilización de recursos	●	●
Mantenibilidad	Modificabilidad	●	●
	Facilidad de análisis	●	●
	Facilidad de pruebas	●	●
Portabilidad	Facilidad de instalación	●	●
	Facilidad de reemplazo		●
	Adaptabilidad		●
	Facilidad de instalación	●	●
Ubicuidad	Transparencia de ubicación		●
	Independencia de implementación		●
	Facilidad de reubicación		●
Delegabilidad	Facilidad de desentendimiento		●
Reusabilidad	Separabilidad		●
	Flexibilidad		●
Manejabilidad	Serviciabilidad		●
	Trazabilidad		●

d. Atributos de calidad de los ESB, definidos en el método ESB-QM

Las características propias de los ESB, enumeradas en el método ESB-QM [6] son:

1 - Ubicuidad

Es el conjunto de atributos de un ESB que brindan a un servicio la capacidad de ser encontrado y utilizado independiente de su ubicación e implementación. Sus subcategorías son:

1. Transparencia de ubicación: Capacidad de un ESB para localizar la instancia de un Servicio independiente de su ubicación. Se logra a través del enrutamiento inteligente y debe poder ser configurado basado en el asunto, contenido o itinerario del mensaje. Con la mediación entre servicios, un servicio cliente que invoque al proveedor de servicio solo necesita saber que el servicio existe. El cliente no necesita saber dónde se está ejecutando el servicio. El ESB localiza el servicio cuando se invoca. Esto proporciona un cierto nivel de virtualización de los servicios, de forma que si un equipo falla, o si se cambia la ubicación de un proveedor de servicio, no es preciso notificar el cambio a cada uno de los clientes individuales. La transparencia de ubicación permite que los servicios se actualicen, muevan o reemplacen sin necesidad de modificar los códigos de las aplicaciones.
2. Independencia de implementación: Capacidad de un ESB para abstraer el contrato de la implementación del servicio.
3. Facilidad de reubicación: Facilidad de un ESB para permitir la reubicación transparente de la implementación de un servicio en caso de que esta haya cambiado.
4. Disponibilidad: Capacidad de un ESB para exponer servicios de negocio tanto a internos como a externos y componer servicios a partir de otros ya existentes a través de lenguajes no procedurales, metalenguajes o cualquier otra metodología.

2 – Delegabilidad

Es el conjunto de atributos de un ESB que le permite a un servicio delegar funciones a otro servicio y recuperar los resultados en forma transparente y confiable a través de éste. Sus subcategorías son:

1. Facilidad de desentendimiento: Capacidad de un ESB para permitir la entrega de un mensaje y liberar de responsabilidad al remitente, esto es logrado a través de la utilización de las facilidades brindadas por la mensajería asíncrona y el paradigma publicación/suscripción.

3 – Reusabilidad

Es la facilidad con la cual aplicaciones existentes o componentes pueden ser reutilizados. Los servicios propios del ESB son modulares y auto-contenidos, reduciendo el número de dependencias de uso entre ellos. Por esta razón, es la capacidad de un servicio propio de ser reutilizado en muchas aplicaciones integradas vía el ESB. Sus subcategorías son:

1. Separability: Es el conjunto de capacidades que permite independencia entre servicios y poca complejidad en sus relaciones. Está asociado con dos atributos: (a) Bajo acoplamiento que refiere a la capacidad de un ESB para permitir la independencia entre las implementaciones de los servicios mediante la definición de funcionalidades acotadas y específicas, y (b) Modularidad, que refiere a la capacidad de un ESB para disminuir las dependencias entre 2 servicios disminuyendo las dependencias artificiales.
2. Flexibilidad: Los ESB facilitan la composición de aplicaciones basándose en servicios, permiten la definición de sistemas complejos distribuidos, incluyendo la integración de diferentes aplicaciones, sistemas, firewall, etc. Además estos pueden

ser construidos a partir de sistemas preconstruidos.

4 – Manejabilidad

Los ESB deben tener las capacidades de administración, de forma que puedan gestionar y monitorear los servicios a los que dan soporte mediante registros y auditorias de servicios centralizados, fallas, estado de procesos, etc. También deben ser capaces de integrarse en sistema de gestión del software. Sus subcategorías son:

1. Serviciabilidad: Capacidad de informar las condiciones de excepción, proporcionando información de las causas y efectos de las mismas.
2. Trazabilidad: Capacidad de informar el paso por cada componente y notificar el estado de una transacción de proceso de negocio en cada uno.

LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

Esta línea de investigación pretende realizar un análisis profundo de las características de los ESB, tanto desde el punto de vista tecnológico como teórico, indicando los distintos escenarios que solucionan y las distintas arquitecturas posibles de ESB para solucionar dichos escenarios. Asimismo se busca aplicar de manera práctica, un método de medición de calidad existente como lo es ESB-QM, a ESB reales como Websphere ESB, propietaria de IBM, y Mule ESB de código abierto, mediante un caso de estudio concreto.

RESULTADOS Y OBJETIVOS

El objetivo del presente trabajo es describir exhaustivamente cómo se debe evaluar un ESB mediante el método ESB-MQ, tanto en sus aspectos funcionales como no funcionales, y realizar la medición de 2 plataformas de referencia, en este caso IBM Websphere ESB de código propietario, y Mule ESB de código abierto. Los resultados giran en torna a la investigación de los distintos patrones de construcción de los ESB, haciendo énfasis en como solucionan efectivamente los distintos problemas de integración de las EAI, SOA y B2B, con bajo acoplamiento y bajo impacto en costos y desarrollo ante los cambios.

Además se estudia y analiza a la metodología ESB-QM, mostrando los distintos aspectos evaluables de un ESB, funcionales y no funcionales, así como las métricas necesarias para realizar dichas evaluaciones.

Por último, se aplica la medición de calidad, mediante un caso de estudio, usando el método ESB-QM, de un ESB propietario IBM Websphere ESB http://www-01.ibm.com/software/integration/wsesb/ y uno open source, Mule ESB http://www.mulesoft.org/.

FORMACIÓN DE RECURSOS HUMANOS

La integración de aplicaciones de software representa un desafío no solamente para la industria sino también dentro de los ámbitos académicos. Su evolución ha dado lugar a la instrumentación de métodos y herramientas que la sustenten e incluso la definición de nuevas arquitecturas como lo es SOA y su principal tecnología de soporte, los ESB. El análisis de calidad, por su parte, es una actividad necesaria para asegurar el cumplimiento de atributos que son propios de los ESB.

El presente trabajo se enmarca en una línea de investigación en SOA-BPM donde se están formando alumnos para desarrollar su tesina e interactuar con docentes e investigadores formados con el objeto de incorporar herramientas de soporte de esta línea de trabajo para solucionar problemas reales.

REFERENCIAS

[1] Thomas Erl – Service-Oriented Architecture: Concepts, Technology, and Design - Prentice Hall – 2005

[2] Juric Matjaz B., Loganathan Ramesh, Poornachandra Sarang, Frank Jennings - SOA Approach to Integration XML, Web services, ESB, and BPEL in real-world SOA projects - Packt Publishing. 2007.

[3] D. Chappell. Enterprise Service Bus. O'Relly Media Inc. (2004)

[4] M. Keen, A. Acharya y et al. IBM Redbooks Patterns: Implementing
an SOA Using an Enterprise Service Bus. (2004)

[5] Nicolai M. Josuttis – SOA in practice – O`Reilly (2007)

[6] Daily Echeverría, Hernán Astudillo, Rodrigo Estrada - ESB-QM: Modelo de Calidad para productos ESBs (2008)

[7] Patricia Seybold Group– Enterprise Service Bus - Evaluation Framework, Criteria for Selecting an Enterprise Service Bus as an Integration Backbone -2005

[8] Ken Vollmer - The Forrester Wave™: Enterprise Service Bus, Q2- 2011

[9] Sanjay P. Ahuja, Amit Patel – Enterprise Service Bus: A Performance Evaluation – 2011