Integrating Cloud Computing Services Using Enterprise Service Bus (ESB)

Liqiang Chen (Corresponding author)
College of Business, Black Hills State University
1200 University St., Spearfish, SD 57799, U.S.A.
Tel: 1-605-642-1268 E-mail: Liqiang.Chen@bhsu.edu

Received: January 2, 2012 Accepted: January 18, 2012 Published: March 1, 2012
doi:10.5430/bmr.v1n1p26 URL: http://dx.doi.org/10.5430/bmr.v1n1p26

Abstract

Cloud computing is evolving as an important IT service platform with its benefits of cost effectiveness and global access. To become a widely adopted IT infrastructure and service platform, cloud computing has to be integrated with other systems in organizations. In academia, there is very limited study of cloud computing integration. In practice, the industry lacks a comprehensive systems integration architecture or tools that can integrate any system universally. Built upon Enterprise Service Bus (ESB) as an integration backbone, this article proposes a universal integration architecture. With this architecture, any system or service (e.g., ERP and cloud computing) can easily be integrated through the ESB without needing software redevelopment. In order to fully support the enterprise-level business operations in a heterogeneous computing environment, this architecture also introduces a rule-based business process management (BPM) engine to streamline business process management across disparate systems.

Keywords: Cloud computing, Enterprise Service Bus (ESB), Business process management (BPM)

1. What cloud computing is and why we use it

Cloud computing (also called utility computing) refers to an IT service model and platform that provides on-demand based IT services over the Internet. Although there are a variety of definitions of cloud computing, the NIST (National Institute of Standards and Technology) definition (version 15) is considered to be the most accurate and comprehensive. According to NIST's definition, cloud computing is composed of five essential characteristics, three service models, and four deployment models. The five essential characteristics are: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured Service (Mell & Grance, 2009). The three service models include:

- SasS (Software as a Service) which delivers software service on demand, such as, salesforce.com – Customer Relationship Management (CRM) service and Google Gmail;
- PaaS (Platform as a Service) which provides the computing platform for companies to deploy and customize business applications on demand, such as, Google App Engine and Microsoft's Azure;
- IaaS (Infrastructure as a Service) which offers data center, infrastructure hardware and software resources on demand, such as, Amazon Elastic Compute Cloud (EC2) and VMware vCloud Datacenter. Both of these resources provide virtual computers for renters to run their business applications.

The four major deployment models include: private cloud, public cloud, community cloud, and hybrid cloud. Companies normally adopt different service models and deployment models depending on their unique business processes and demands on IT services.

Cloud computing today is an evolution and application of modern ICT including server virtualization, autonomic computing, grid computing, server farm, network storage, and web service (Buyya et al., 2009; Marston et al., 2011). Cloud computing has many advantages. For example, it saves on IT spending, reduces new systems' implementation time, efforts and risks, eliminates regular system maintenance (e.g., software and security updates, systems and data backup, system recovery), and provides pervasive IT services. However, cloud computing also has some issues and pitfalls. For example, the cloud computing market lacks legal protection and regulation (Buyya et al., 2009; Marston et al., 2011); business users have concerns on security, privacy, losing control of data and services, and being locked-in by service providers (Buyya et al., 2009; Machado et al., 2009; Marston et al., 2011). In addition, it is difficult to integrate cloud computing with other systems and IT services; particularly, the IT industry lacks standardized cloud computing interfaces, which further prevents cloud computing integration (Machado et al., 2009). Lack of a high-level integration

capability has prohibited cloud computing from being a widely adopted IT service platform (Curran, 2009). Too many "clouds" in a company would mean rainy days in its business.

2. Integration is the key to adoption of cloud computing and business success

Due to the high complexity and risk in the integration of cloud-to-cloud and cloud-to-on-premise systems, most cloud computing services (i.e., SaaS) currently are limited to general-purpose applications and standalone services (Marston et al., 2011). Most clients of SaaS clouds are SMEs (Small and Medium Enterprises) because their business processes tend to be more commonly shared and relatively simpler than those in large companies. Additionally, "having much less of legacy IS infrastructure to contend with, it will also be much easier for SMEs to move to the cloud (and in many cases, the cloud might be the first instance when they try a new functionality, e.g. ERP, because the traditional alternative would have been too expensive in the first place)" (Marston et al., 2011). By contrast, many large companies run their business processes on either legacy or on-premise systems or both in a heterogeneous computing environment. Their business processes cannot easily migrate to cloud computing. According to a study conducted by Forrester Research, one of the key hurdles and challenges in SAP moving forward into cloud computing is that SAP software suite as a whole has not been optimized for the cloud yet, and SAP "still presents a fragmented approach and is clearly in no hurry to make its full enterprise suite available from the cloud" (Martorelli & Herbert, 2010). Enterprises will gradually adopt cloud computing and will need a powerful tool to integrate their on-premise systems to clouds for the short term. For the long term, they need a fully integrated infrastructure of cloud-to-cloud after their business processes are completely on clouds. Unfortunately, system integration is still limited to a few types of systems and a small scale of IT services (Marston et al., 2011). Without a comprehensive system integration architecture or tools, cloud computing will not be widely adopted, nor will it become a fundamental business infrastructure.

Nowadays, business processes in organizations are transformed in a fast pace to adapt to the highly dynamic and turbulent business environment. Moreover, "enterprise service consumers with global operations require faster response time, and thus save time by distributing workload requests to multiple Clouds in various locations at the same time" (Buyya et al., 2009). This requires companies to be able to integrate many business solutions in a heterogeneous computing environment in a timely manner. The system integration thus becomes one of the major requirements and objectives in managing business processes, and it is critical to business success (Bhatt, 2000). The business community needs a configurable, scalable, and highly interconnected IT service platform. However, there are many challenges involved in creating dynamically interconnecting and provisioning among various systems within and across enterprises (Buyya et al., 2009). Fortunately, with the latest information technologies, we are able to design a universal integration architecture for various information systems and services.

3. ESB-based architecture for cloud computing integration

With focus on the system integration between SaaS clouds and other systems that implement and manage business processes in organizations, the article proposes a high-level and fully-implementable architecture for system integration between cloud computing and other systems. The architecture is built upon ESB (Enterprise Service Bus), a Service-Oriented Architecture (SOA) technology. In order to handle business process management (BPM), which is a main purpose of system integration, this article also introduces a rule-based business process management (BPM) engine in the architecture.

3.1 What ESB is and why we need it

Integrating disparate systems in a heterogeneous computing environment has been a huge challenge and a high risk in the business community (Bernstein & Haas, 2008). But, it is critical to business success particularly for automating and streamlining business process management (BPM). API (Application Programming Interface) has been a major method to integrate two different systems or IT services. API method is tightly-coupled and only works for one-to-one connection. API is primarily used for data exchange between systems and lacks security management. Moreover, the industry lacks standards for API due to systems being greatly different and proprietary.

In the late 1990's, the IT industry designed and developed a Service-Oriented Architecture (SOA) aimed at creating a distributed computing architecture in which all software services could be integrated. In SOA, software services are distributed on networks and they are integrated with each other via a central service registry which is called broker (Lawler & Howell-Barber, 2007). When a software service needs another service, it queries the registry with certain criteria. If the registry finds the service that matches the criteria, it sends a service contract with an endpoint address back to the requester. The requester then remotely invokes the requested services with the granted contract and address. When a new service is plugged into the SOA network, it registers itself in the central service registry for future requests. A SOA service is self-contained to implement certain predefined business logics; that is, the service interface

development is completely separated from the business logic implementation in the SOA service. A requesting service never needs to know how the requested services are executed and what protocols and data formats are used. In contrast to the tightly-coupled API method, SOA is truly loosely-coupled software service architecture (Lawler & Howell-Barber, 2007). With the open standard service registry and contract mechanism, SOA makes system integration into a plug-and-play process without requiring software redevelopment. In addition, SOA services implement a quality of service (QoS) associated with them, including security management, reliability assurance, and service-related policy management. In general, SOA, as technology-neutral software architecture, provides reusable, flexible, scalable, and distributed software service platform with a high industrial standard.

In the early 2000's, the IT industry introduced a bus-architectural technology, Enterprise Service Bus (ESB) (Chappell, 2004). ESB greatly improves the central service registry mechanism in SOA and provides a software infrastructure for SOA implementation in enterprise applications as well as system integration across organizational boundary. That is, ESB greatly enhances the usage of SOA with a virtual bus to connect many disparate systems and services together (Chappell, 2004; Schmidt et al., 2005; Menge, 2007; Bygstad & Aanby, 2010). In particular, ESB is message-based bus architecture which consists of a set of software components called service containers (Menge, 2007; Bygstad & Aanby, 2010). Service containers are interconnected over a reliable and secure messaging channel. A service container connects one or many software services or systems through service adapter(s) (Menge, 2007). A system or service sends request messages directly to its connected service container which in turn processes and routes the messages to a destination (requested) service container. The destination service container processes and forwards the messages to its connected destination (requested) service or system through a service adapter. During this process, ESB service containers process, monitor, log and manage messages to make sure all services and systems are connected reliably and securely. Figure 1 depicts the architecture of ESB. A service container contains adapter module, mediation module, message routing module, security module, and management module. Each module is responsible for specific tasks.

The **adapter module** acts as a service connector to connect various software services and systems such as ERP (e.g. SAP, Oracle E-Business Suite, Microsoft Dynamics) or CRM (e.g. Salesforces.com). A service adapter, very similar to a hardware or software driver (e.g. printer driver, database driver), uses the native transaction interfaces in the proprietary service to transfer messages between the service container and the service. Normally, one service adapter is required for a proprietary system or service. In the market, ESB solution providers (e.g., IBM, Oracle, MuleSoft, SAP) provide a range of service adapters for various systems and services.

The **mediation module** transforms protocol, message format, and message content between the requesting service and the service provider. Mediation is critically important for the system integration because services on the ESB use different protocols and data formats (Schmidt et al., 2005; Menge, 2007). The mediation module can be implemented using the XML-based transformer components which are configured through XSL (Extensible Stylesheet Language). More complex transformers are required to invoke other SOA services or query database (Menge, 2007).

The **message processing module** processes incoming and outgoing messages and implements event handling. This module sorts, prioritizes, delays, and reschedules message delivery as needed to guarantee both synchronous and asynchronous communications. It monitors and logs messages for quality of services (QoS) and security management (Schmidt et al., 2005). Using the message-encoded logic and content-based logic, the module conducts message validation, transformation and aggregation, and message buffering and delaying. In addition, the module supports various event handlings such as event triggering, noticing, filtering, and mapping (Menge, 2007). Artificial intelligence technologies can be built in this model to improve message processing and event handling. For example, in the ESB of an investment bank, a proactive event handler automatically checks a stock market and updates the data correspondingly so that the future event handling always uses the latest market information.

The **message routing module** routes messages from a service requester to the service providers using the XML-enabled content-based routing method (Menge, 2007). The content-based method provides a highly configurable, dynamic and intelligent routing (Menge, 2007). For example, when the message content indicates a customer doesn't want to book a flight ticket, the request will not be routed to the service provider that books a flight ticket. The routing module is implemented with popular web service techniques such as Simple Object Access Protocol (SOAP) and XML Path Language (XPath). The routing module provides a reliable and secure message exchange channel between the service requester and the service provider.

The **security module** enforces compatibility between all modules and security policies. Particularly, the security module implements security standards and policies including authentication, authorization, encryption, auditing, and intrusion detection. The security module is also responsible for unifying security management throughout the service container during message receiving, processing, and sending (Menge, 2007).

The **management module** plays a critical role in ESB. This module tracks activities happening in a service container and handles a variety of exceptions. It also manages workload, schedules tasks, creates threads, registers services, manages service transaction and its lifecycle, and manages service state and quality of service (QoS). ESB is a virtual service bus architecture that connects many distributed and decentralized service containers (Schmidt et al., 2005). Additionally, service registration and invocation are highly dynamic. This requires ESB to be highly configurable and customized to meet different service demands. The management module therefore provides administration tools that configure, manage and control service container. For example, system managers use administration tools to add security polices according to the latest security updates. A centralized management tool can be built to manage and configure all service containers in the ESB.

In sum, ESB is message-based distributed integration software platform in SOA (Menge, 2007). It is open-standard, platform-independent and vendor-neutral. It can run on any operating system and hardware structure, and can be implemented with different technologies (e.g. J2EE, Microsoft .NET). With many service containers distributed and decentralized on the Internet, ESB creates a virtual service bus for system and service integration (Schmidt et al., 2005). With administration tools, users can configure ESB containers without requiring shutdown or interrupting the integrated services. ESB adopts SOA and highly enhances the SOA implementation and functionalities by replacing the central registry with the bus architecture. It makes the system and service integration a truly plug-and-play process. In the market, major ESB vendors include IBM, Oracle, MuleSoft (Open Source), Progress Software, Software AG, and Red Hat. With currently available ESB solutions, integrating cloud computing with other systems can be fully implemented in the following integration architecture.

3.2 Fully implementable ESB-based integration architecture

Business processes management (BPM) requires a comprehensive integration among a variety of systems and services. The ESB technology can fully integrate disparate systems and services together without requiring system or service redevelopment. Using ESB as an integration backbone, a BPM engine manages workflow among systems and services. The fully-implementable integration architecture that streamlines BPM is proposed as shown in Figure 2.

The BPM engine is a software service that implements BPM. A business process consists of many workflows which are a series of interrelated tasks with data flowing and processing. It can be a process of person-to-person, person-to-system, system-to-system, or a combination of the three. There are three general business processes in organizations: operational, management, and support. Many business processes in an organization are interrelated and work together to perform enterprise operations. There are two dominant BPM approaches: the graph-based approach and the rule-based approach (Lu & Sadiq, 2007). The proposed architecture in this article adopts the rule-based BPM engine to automate and streamline business processes distributed in a heterogeneous computing environment.

The rule-based BPM engine has many advantages. For example, rules can express more workflow patterns than graph-based languages; exceptions can be easily handled with rules; rules can be dynamically changed to realize ad hoc processes, and rules are flexible to execute interdependent tasks "without explicit specification on the conditions for parallel or serial execution" (Lu & Sadiq, 2007). In addition, best practices can easily be implemented in rules, and fast developing artificial intelligence technologies help generate many smart rules. Using administration tools, business managers can easily reconfigure rules to manage business processes dynamically and implement synchronous (e.g. system-to-system) and asynchronous (e.g. person-to-person, person-to-system) BPM ad hoc.

In Figure 2, the ruled-based BPM engine handles all business workflow management using the predefined rules in the "Business Rule Repository" database. Using the "Management & Analytical Tool" GUI (Graphic User Interface), managers not only dynamically manage business processes by manipulating rules, but also perform process modeling and simulating as well as process analysis and design. On the "Unified Service Interface" GUI, users manage and monitor business processes. In general, the rule-based BPM engine acts as a controller or brain in the integration architecture to conduct various critical tasks such as managing, controlling, automating and optimizing workflows among distributed systems.

The proposed architecture fully separates BPM from the integration backbone - ESB. This architecture is more flexible, robust, scalable, and portable. Like any other system or service, the rule-based BPM engine is connected to the ESB via the adapter in a service container. Physically, service containers can be distributed on one or several enterprise middleware servers. The virtual bus is established via secure communication protocols (e.g. HTTPS, SSH) among service containers. In contrast to a central BPM engine as proposed in this article, the BPM tasks or rules can be divided and distributed into many BPM agents running on middleware. These BPM agents all together are considered to be an application layer on top of the ESB, and they can be implemented with open source EJB (Enterprise Java Bean) or Microsoft .NET. In this decentralized BPM architecture, the rules are built in each BPM agent and workflow is realized with a series of tasks executed by BPM agents in a predefined sequential order. The predefined sequential order is coded

in the request messages so that the ESB service container which receives the request knows who the next BPM agent is by using the content-based routing method. MuleSoft adopts this design in its ESB-based BPM architecture (Menge, 2007).

It is noteworthy that recently the XML-based Business Process Execution Language (BPEL) is emerging as a BPM tool within web services. BPEL is used to orchestrate business processes distributed in web services. Compared to a general purpose rule-based BPM engine, BPEL is light-weight and open standard. BPEL implements business process management without requiring ESB as integration backbone. However, BPEL only works with web-based business solutions. In addition, BPEL provides only limited business decision logics (Paschke & Kozlenkov, 2008). Accordingly, BPEL can be used to build a simple web-based BPM platform. However, it won't replace the ruled-based BPM engine which manages large scale and highly complicated business processes in a distributed and decentralized computing environment.

4. Conclusion

This article proposes a high-level comprehensive architecture for system and service integration. Built on ESB as backbone, the architecture introduces the rule-based BPM engine to automate and streamline business process management in organizations. The proposed architecture is fully implemented with ESB solutions currently available on the market. There are increasing demands on out-of-box integration of business processes across organizational boundary even though most companies still run their business process management on their on-premise systems. With more business processes migrating to cloud computing, enterprises need highly integrated infrastructure to manage their business processes globally. Therefore, future research and practice will be focused on developing service adapters that connect various systems and services to the ESB backbone. Furthermore, designing and developing a highly intelligent and self-adapted BPM engine on the ESB backbone become critically important for the future business success, and this should be put on the highest priority in both academia and the industry.

The proposed architecture can also serve as an architectural blueprint for future system integration with emerging technologies. With technology convergence, many business services have been created and/or moved onto a mobile platform (e.g. tablet PC, smartphone). Mobile systems have been become an important component of the IT service platform today. With appropriate service adapters, mobile computing services can be plugged to the ESB backbone in the integration architecture. ESB provides business users with a transparent integration platform without distinguishing systems or technologies that deliver business services (Schmidt et al., 2005). The unique feature of context-based services on mobile systems greatly enhances business services and process management in organizations. The future BPM engine will integrate and utilize the context-based functions to provide highly satisfactory services for customers. In summary, building on the ESB technology, future business process management will seek a more intelligent and adaptive BPM engine to create high quality services in a dynamic business environment. The proposed integration architecture can serve as a foundation for streamlining BMP across multiple disparate platforms including cloud computing, ERP, and other on-premise systems.

References

Bernstein, P. A., & Haas, L. M. (2008). Information integration in the enterprise. *Communications of the ACM, 51*(9), 72-79. http://dx.doi.org/10.1145/1378727.1378745

Bhatt, G. D. (2000). An empirical examination of the effects of information systems integration on business process improvement. *International Journal of Operations and Production Management, 20*(11/12), 1331-1359. http://dx.doi.org/10.1108/01443570010348280

Bygstad, B., & Aanby, H. (2010). ICT infrastructure for innovation: A case study of the enterprise service bus approach. *Information Systems Frontiers, 12*(4), 257-265. http://dx.doi.org/10.1007/s10796-009-9169-9

Buyya, R. R, Yeo, C. S., Venugopala, S., Broberga, J., & Brandicc, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599-616. http://dx.doi.org/10.1016/j.future.2008.12.001

Chappell, D. A. (2004). *Enterprise Service Bus*. O'Reilly Media Inc.

Curran, C. (2009). The Biggest Barrier to Cloud Adoption (2009). [Online] Available: http://www.ciodashboard.com/cloud-computing/cloud-adoption-barrier/

Lawler, J. P., & Howell-Barber, H. (2007). *Service-Oriented Architecture: SOA Strategy, Methodology, and Technology*. Boca Raton, FL: Auerbach Publications, Taylor & Francis Group. http://dx.doi.org/10.1201/9781420045017

Lu, R., & Sadiq, S. (2007). A survey of comparative business process modeling approaches. In *Proceedings of the 10th International Conference on Business Information Systems* (pp. 82-94). Springer-Verlag Berlin, Heidelberg.

Machado, G. S., Hausheer, D., & Stiller, B. (2009). Considerations on the interoperability of and between cloud computing standards. In *27th Open Grid Forum (OGF27), G2C-Net Workshop: From Grid to Cloud Networks*, Canada.

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing - The business perspective. *Decision Support Systems, 51*(1), 176-189. http://dx.doi.org/10.1016/j.dss.2010.12.006

Martorelli, B., & Herbert, L. (2010). Cloud computing offers both near-term and long-term benefits for SAP customers. Forrester Research Inc. [Online] Available: http://www.forrester.com/rb/research/

Mell, P., & Grance, T. (2009). The NIST Definition of Cloud Computing (2009). [Online] Available: http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf.

Menge, F. (2009). Enterprise Service Bus. In *Proceedings of Free and Open Source Software Conference.* Germany.

Paschke, A., & Kozlenkov, A. (2008). A Rule-based Middleware for Business Process Execution. In *Proceedings of Multikonferenz Wirtschaftsinformatik* (pp. 1409-1420). Germany.

Schmidt, M. T., Hutchison, B., Lambros, P., & Phippen, R. (2005). The Enterprise Service Bus: Making service-oriented architecture real. *IBM Systems Journal, 44*(4), 787-791. http://dx.doi.org/10.1147/sj.444.0781

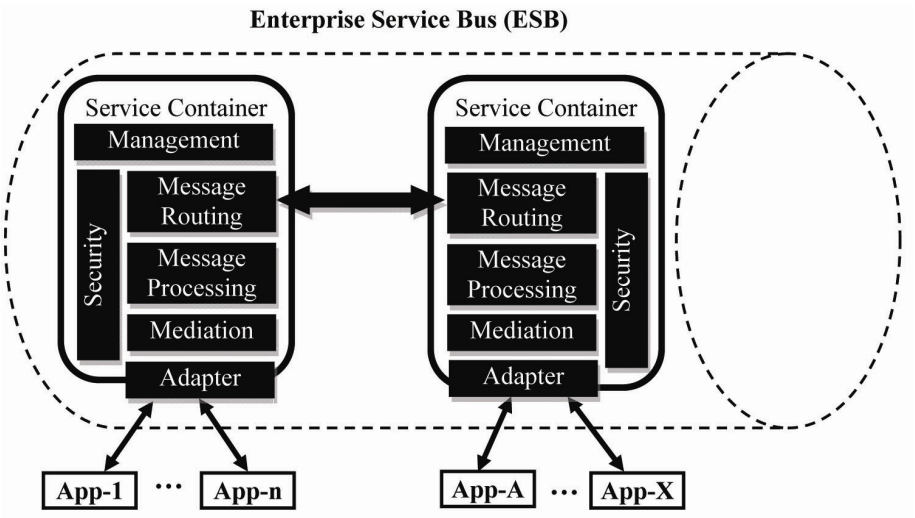

Figure 1. The Architecture of Enterprise Service Bus (ESB)

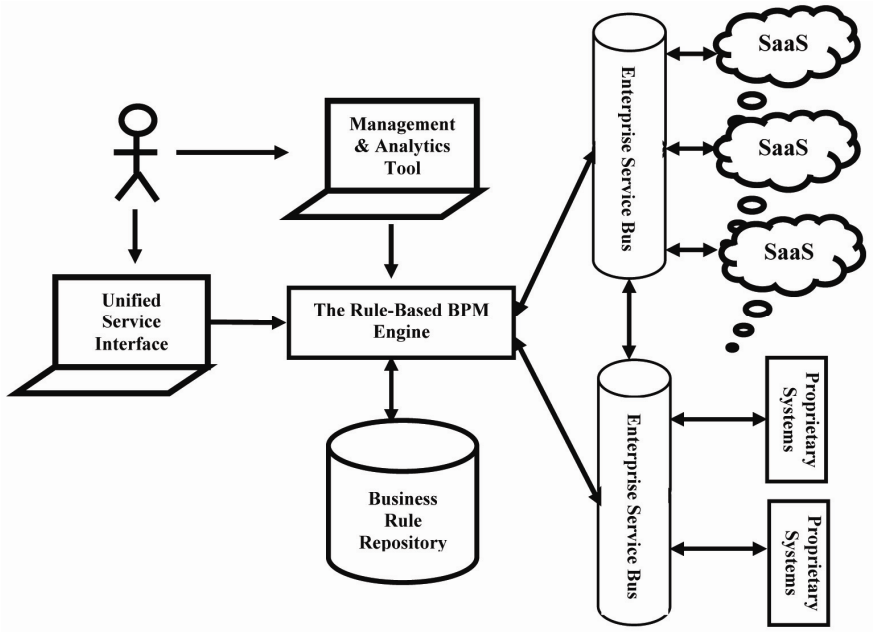

Figure 2. The ESB-Based Integration Architecture