Open Source ESBs for Application Integration (SOA Optional)

by Anne Zelenka

Lightweight open source enterprise service bus (ESB) implementations offer a low cost, scalable, and practical approach to enterprise application integration. Some commercial vendors offer the idea that ESBs must be the heart or backbone of an SOA and therefore require a significant investment of money along with additional integrated components. But you can achieve high-performance application integration without buying into an entire SOA stack and story.

ESBs don't have to be about SOA, but if you are interested in moving towards a service-based architecture while integrating disparate applications, you may want to experiment with one as a way of moving incrementally in that direction. Open source offerings provide an ideal way to do that because they have no license fees, allow you to modify and extend the code as you see fit, and may be supported by thriving developer and user communities focused on practical use.

What is an ESB?

Although there is some dispute as to what an enterprise service bus includes and what qualifies as one, an implementation of the ESB architectural pattern usually displays these characteristics:

An ESB establishes a unified communications channel and uses a bus architecture for scalability.

- **Uses a bus architecture for scalability and reliability**. The first generation of enterprise application integration (EAI) platforms that became popular in the late nineties usually used a central broker through which all application communication traffic passed. A bus architecture, on the other hand, distributes its processing to where the applications reside rather than centralizing it at a hub.

- **Establishes a unified communications channel**. This is often implemented using a messaging server such as JMS. Different implementations of the ESB architectural pattern may require the use of certain message formats such as the WS-* XML-based web services specifications.

- Provides **integration, mediation, and communication services** that include logging application events, transforming messages from one format to another, routing messages based on content or other factors, guaranteeing delivery, and enforcing security requirements.

- **Standards-based**. This is another way in which ESBs differ from their EAI forerunners, which generally used proprietary communications formats and protocols. For example, most ESBs “understand” the WS-* set of interface specifications. Some ESBs follow the Java Business Integration specification (JBI) which is a WS-* compliant description of an SOA infrastructure while others adhere to the Service Component Architecture (SCA).

- May provide **additional features** such as process orchestration, a rule execution engine, or automated service discovery.

ESBs and Service-Oriented Architecture

ESBs don’t have to be used in the context of a grand SOA effort and similarly, implementing an SOA doesn’t require an ESB. Furthermore, neither an SOA nor an ESB requires the usage of a particular set of standards like WS-* or JBI. Neither an SOA nor an ESB supposes a particular set of technology components. SOA-style thinking favors flexibility over one way of doing things.

ESBs don’t have to be used in the context of a grand SOA effort.

What is SOA, if not a WS-*, SOAP-based enterprise architecture? Perhaps it is more usefully thought of as an architectural pattern for enterprise information technology in which coarse-grained application services are combined and recombined flexibly to meet the needs of an ever-changing business. SOAs almost always integrate disparate applications and technologies, and ESBs are often used to achieve this integration.

The ultimate goal of SOA is a set of application processes driven by business needs rather than the other way around. This doesn’t mean, therefore, that an SOA infrastructure should be designed from the top down taking the current business requirements as forever static. In fact, the most practical way to achieve that goal of technology subordinate to the business may be to build the infrastructure itself using SOA-style thinking. That is, compose it from reusable and interoperable pieces that you can mix and match, add and discard, modify and expand, as the business needs and requirements inevitably change.

Commercial ESB Implementations

The big enterprise software vendors sell ESBs, along with entire stacks of SOA infrastructure. Oracle offers an ESB as part of its SOA Suite, which includes a BPEL process manager, a business rules engine, and a web services registry, among other components. BEA markets its AquaLogic Service Bus alongside its WebLogic Server as an "enterprise-class integration backbone." IBM's WebSphere can be outfitted with an ESB from IBM that can serve as the "heart of your SOA."

Smaller software vendors like TIBCO and Progress also sell ESB implementations. TIBCO markets its ESB as part of its BusinessWorks "Integrated Services Environment." Progress puts its Sonic ESB at the foundation of its product family, which includes an orchestration server, XML server, and database service.

Beyond offering integrated and comprehensive SOA platforms, some commercial vendors and their consultants suggest an even bigger SOA story–the business story–where SOA becomes not just about service-orienting your IT, but also service-orienting your way of working. Sometimes this is called shared services. It allows business functions like HR or purchasing to be decentralized and recentralized in different ways, to achieve cost savings or strategic objectives or ideally, both.

Current Open Source ESB Offerings

But ESBs don't have to be primarily about service orientation and they don't have to be aligned with your business in order to be worth an investment of some time and experimentation. If you are concerned primarily with legacy application integration and with exploring loosely coupled architectures, experimenting and prototyping with open source ESBs might be a productive path to take.

A company contemplating an incremental and practical bottom-up approach to application integration might want to try out the standalone, lighter-weight open source ESBs. These are enterprise service buses that are not always delivered as part of an SOA platform or stack but are rather components that may interoperate with what you already have. Current open source ESB offerings include Mule, Synapse, Celtix, ServiceMix, JBossESB, and Open ESB.

Companies are already using open source ESBs productively.

Open Source ESBs in Use

Although the ESB category is relatively new in the open source landscape, companies are already using them productively. For example, Mule ESB users include European direct debit and credit processor Voca, financial services giant State Street, mega-retailer Wal-Mart, and Japan's Osaka Gas and Information Systems. Atkins Global, an engineering consultancy and design firm, has selected IONA's Celtix to quickly begin an SOA initiative while containing upfront licensing costs. Apache ServiceMix is being deployed by a large government agency in a system supporting clinical trials reporting, has been embedded within a commercial data integration solution, and is used as the platform for a distributed energy trading and risk management solution.

A few actual usage scenarios will illustrate the practical, bottom-up approach just described. These scenarios suggest how companies can achieve integration success at the same time they experiment with SOA concepts and tools.

Developing Application Integration Strategy for Aircraft Maintenance IT

A major US airline is experimenting with a lightweight open source ESB implementation in order to develop a strategy for integrating a large number of applications supporting aircraft maintenance. The applications requiring integration include custom-built mainframe systems, Teradata warehouses, and SAP ERP applications. Operating systems in use include z/OS, Unix, and Windows; programming languages include J2EE and C++. The J2EE applications are considered legacy applications and were built without service orientation in mind.

A lightweight standalone open source ESB was selected for this ongoing prototyping project because of its clean, easy-to-understand architecture and for its JBI support. JBI support was considered important by the team in the hopes it might prevent proprietary lock-in in the future.

Creating a Web Gateway for an Existing Trading Platform

A financial services company extended its trading platform with web services access by creating a bridge server using an open source ESB. The bridge server handles data transformation and provides for scalability horizontally (i.e., increased traffic) and vertically (i.e., additional functionality). The lack of license fees was a main factor in the decision to choose an open source ESB.

The team found the documentation and tooling in the open source ESB to be somewhat lacking.

The company factored in the ESB's usage of standards. Their preference was not to be tied to JBI, so an ESB that didn't require the use of JBI was favored. JBI adds complexity with features their team didn't need, such as dynamic discovery of services.

Selecting an open source ESB wasn't without tradeoffs. The team found the documentation and tooling in the open source ESB somewhat lacking. They would like to see additional documentation about integrating with various application servers and about getting started. They'd also prefer to use a graphical tool for creating configurations rather than writing XML directly.

A Risk Management Company Chooses a Lightweight ESB

A risk management company chose an open source ESB because it could work with any web application server including WebLogic, WebSphere, and Oracle's OC4J. Although they have only used the selected ESB internally so far, the possibility of expanding their web services infrastructure to clients in the future was of major importance.

The company perceived TIBCO's offering as a bit too heavy and requiring too much infrastructure to set up before they could start. They favored an ESB that worked well with the Spring framework. They didn't take on a comprehensive comparison of ESBs because they needed to get something working quickly.

Evaluating Open Source ESBs

Not all enterprise service buses are created equal and the same holds true within the category of open source ESBs. Among the considerations you'll want to take into account when evaluating an ESB are these:

- **Origin**: where and how a project originated matters, but not in any deterministic way. Projects started by developers in order to meet a defined technical need and projects begun by for-profit companies as a way of expanding their business model can produce equally viable open source efforts. By understanding the history of a particular ESB implementation, you will be better equipped to judge how it might fit your organization's needs.

- **Maturity**: an ESB that's still in an alpha or beta stage will likely have fewer features and perhaps more bugs than one that's gone through more releases. On the other side, a less mature one might give your organization more opportunity to participate in its development and evolve it in the direction that works with your long-term plans.

- **Level of commitment to standards**: certain projects are built from the ground up in order to conform to and support certain standards (e.g., ServiceMix with JBI and Synapse with the WS-* specifications). Others, like Mule, are more agnostic with respect to standards, supporting them but not dogmatically. You may want to choose your ESB based on your own level of commitment to standards.

- **Flexible deployment options**: if you are interested in experimenting with an open source ESB, you may want the option of deploying initially in a very simple manner so you can get started quickly. Some of the ESBs support a variety of deployment models that can allow you to start simply and then move to more advanced models as your requirements demand.

One major benefit of open source can be a thriving community providing assistance and new features.

- **Platform support**: you'll of course want to consider whether the ESB you're thinking of using supports the platforms (application servers, web servers, messaging middleware, application frameworks) you're already using.

- **Community viability and momentum**: one major benefit of open source can be a thriving community providing assistance and new features, but not every open source project creates a sustainable ecosystem. Check the developer forums, the product roadmap, and any mailing lists to get a quick read on the community supporting the project.

- **Commercial support:** commercially available support represents an important commitment on the part of a business—and for many IT shops, lack of commercial support would remove a product from consideration. You need more than just a way of logging bugs and getting patches. Look for performance tuning and training support. Also, check whether the company providing support has commit rights on the project. If they don't, they may not be able to provide you with the critical fixes you need on a timely basis.

- **Tooling and documentation**: open source projects are often thought of as "by developers for developers" and this has some truth to it. Open source ESBs may have rudimentary tooling and documentation compared to their commercial counterparts. If your development staff has the ability to figure things out without documentation and GUIs, then this won't matter as much.

RedMonk Red Lights

Increasingly, the most popular business model for software is a hybrid that combines open source and commercial licensing models. Drawing a too-thick line between open source and commercial ESBs therefore would be misleading. Most viable open source projects have some commercial entity backing them up in some way (MuleSource for Mule, LogicBlaze for ServiceMix, and IONA for Celtix, for example). On the other side, medium-sized and large commercial software vendors are experimenting with how open source efforts fit in with their own plans. For example, Sun offers Open ESB as an alternative to its Java CAPS platform.

You don't have to buy into an entire SOA story or expensive SOA stack to benefit from ESBs for application integration.

Another reason to resist any strict categorization of open source versus commercial ESBs is that open source ESBs are not necessarily lightweight and standalone while commercial offerings may be able to be deployed that way. Some open source ESBs are available as part of an integrated SOA stack, and some commercial options may work well as a drop-in component, even though they're not always marketed that way. If you do want to experiment, the easiest course is certainly an open source ESB, because you can download the source and go, no purchase order required.

Be aware of increasing disenchantment percolating throughout IT with SOAP and the WS-* specifications. Though SOA is often considered inextricably linked to these standards, some practitioners feel that they are architecturally counter to the loosely coupled, URI-based, RESTful nature of the web itself. It may be prudent to take a wait and see approach on such standards and choose a product that can support them but doesn't require or assume them.

RedMonk Take

Open source ESBs represent a potentially useful tool for enterprise application integration, and you don't have to buy into an entire SOA story or expensive SOA stack to benefit from them. Whether you resonate with SOA-style thinking or not, you may want to take one for a test run to see how it can help you integrate your legacy and new applications. Their flexible deployment models, low barriers to entry, and community support make them ideal for experimenting with new ways of achieving loosely coupled application integration.

About RedMonk

RedMonk is a research and advisory services firm that assists enterprises, vendors, and systems integrators in the decision-making process around technology deployments. We cover the industry by focusing on business and operational context rather than speeds and feeds and feature tick-lists.

Founded by James Governor and Stephen O'Grady, and headquartered in Denver, Colorado, RedMonk is on the web at www.redmonk.com.