Key Transformational Techniques to Achieve Enterprise-Scale Interoperability

Shamlan Siddiqi
Keane, Inc.

This article examines key modernization and transformation strategies for interoperability, including enterprise use of open source, service-oriented architecture (SOA), and agile techniques in software development. The article concludes with a real-world case study on legacy modernization and interoperability for a major government agency through use of these tools and techniques.

Modern concepts such as SOA and principles around Web 2.0 and Web 3.0 rely on interoperability as their foundation. This is different from the traditional view of integration where one or more adapters are required to glue things together. The intelligent use of standard interfaces between multiple components, leveraging open source technologies, sophisticated SOA design principles, and a flexible software development methodology can be a much more cost-effective, efficient, and scalable option for the enterprise interoperability of systems.

While the adoption of open source technologies for interoperability has been growing at a fast pace over the last few years, I find that there is still apprehension in the *enterprise use* of open source as the dominant platform in the interoperability of mission-critical systems. By *enterprise use*, I mean using a complete stack of non-proprietary open source tools for everything including a back-end database at the data layer, core applications and modules at the application layer, a messaging engine/service bus or standard interface strategy at the integration layer, and creative user interface design techniques leveraging advances in the Web 2.0 space. That being said, open source software by its very nature requires that its software code be open, extensible, and openly available. This paradigm enables programmers globally to share thoughts and ideas, constantly increasing the power and quality of development tools and large applications. Additionally, I believe that the rapid adoption of SOA design principles is a significant boost to the *modernization* of interoperable open source tools. Of course, this rapid evolution of sorts creates problems.

Issues With Interoperability and Security in the Open Source Arena

With new tools and technologies being released often, agencies face challenges in determining whether a particular tool will work with an existing legacy component, whether the operating system is compatible, and if the database will work. There are also the myriad complexities that come with defining the proper interface to create reusability. Some major interoperability issues include centralized identity management, data integration including real-time data synchronization, batch transfer, and portability. This is critical as organizations want their solutions to work across different platforms, such as the various Linux options as well as Windows.

Other issues include the perception of security (or lack thereof) with open source tools as well as the trustworthiness of the data. It is easy for developers to be able to add malicious code or add functions (etc.) to source code they have downloaded. Both of these challenges can be mitigated by the appropriate selection of open source projects. It is advisable to select projects that are more established, come from a reliable source, and have built-in security mechanisms rather than ones that may not have huge support or documentation. Typically, source code from reliable projects does not usually have too many security issues because they are controlled by an open source committee or group. Many development projects use open source tools and technologies and provide ongoing maintenance and support on those projects as well. In fact, based on cases I have encountered, support and maintenance issues are resolved more rapidly with open source because a developer can go into the code and make the fix. The alternative—waiting for a product vendor to come in, assess the problem, and make adjustments—usually takes a significant amount of valuable development time and money.

> *"This paradigm enables programmers globally to share thoughts and ideas, constantly increasing the power and quality of development tools and large applications."*

Interoperability Using an SOA Design and Enabled by Web Services (WS) and the WS* Protocols

Using the SOA design paradigm, interoperability is typically accomplished by developing WS using industry standard programming languages such as eXtensible Markup Language (XML), Web Services Description Language (WSDL), and others. These basic WS standards have evolved over the last few years and the Web Services Interoperability Organization (WS-I)—an open industry organization chartered to establish best practices for WS interoperability—has released Basic Profile (BP), containing implementation guidelines for basic WS standards. Many open source tools are starting to adopt these standards using guidance from BP as part of their ongoing product road map. I think this is a critical success criterion in the further evolution and adoption of open source tools for enterprise integration and other enterprise-level functions. However, just following WS-I standards and guidelines is not sufficient enough to achieve interoperability.

Best Practices in Implementing WS-Enabled SOA Interoperability

It is important to note that the WS*-based development can be complex. Based on experience (and as typically in many things), an incremental strategy is the most effective one. Start implementing using only the basic WS specification as opposed to multiple WS* protocols, which can add more complexity to development. This will create problems when trou-

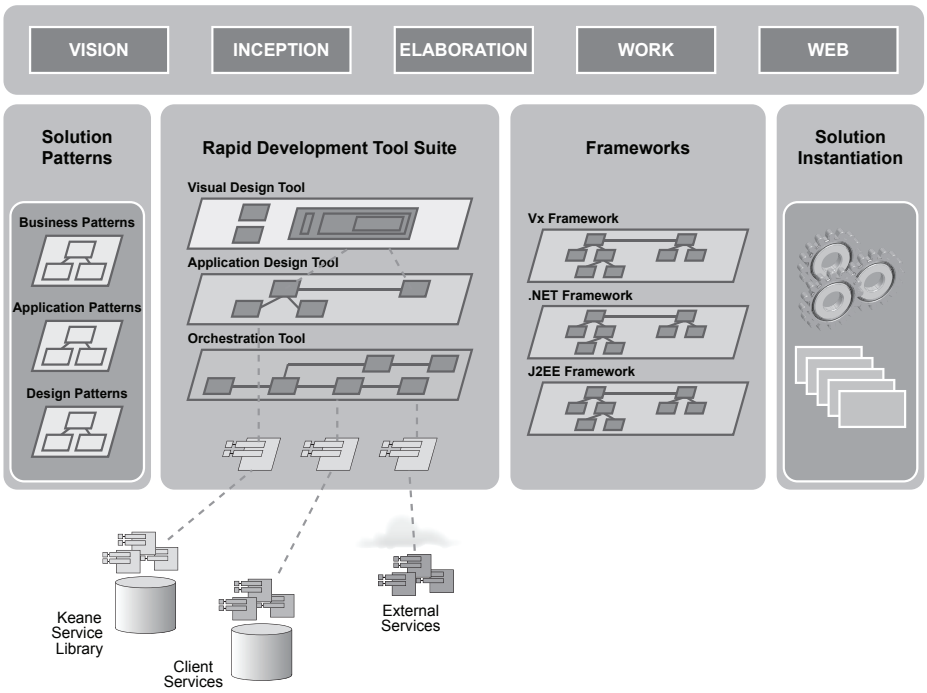

Figure 1: *Agile-RUP Methodology*

bleshooting. Many government organizations are facing major development hurdles because of large-scale implementations of WS without using a chunking strategy. These days, vendors provide BP-compliant products and have tested those products comprehensively for interoperability. It is important to follow these guidelines and use these tools. The open source community (as previously mentioned) is adopting these specifications for interoperability as well. Additionally, it is always important to seek advice from multiple forums and blogs, as there are people who may have encountered similar issues.

The BP consists of guidelines or best practices recommending how the mentioned specifications (XML, WSDL, etc.) should be used together to develop interoperable WS. The guidelines cover messaging, description, discovery, and security. Some key Web service specifications covered in the initial release (BP 1.0) include common standards such as XML 1.0 and WSDL 1.1[1].

When discussing the importance of interoperability and standards, I like using the example of Boeing. When taking apart old 747s, Boeing reuses a lot of the core parts (recovering more than $6.8 million) due to the use of common standards in the design of the original Boeing 747. However, specific standards must be agreed upon before this reuse can take place, and this is not applicable to other jets. Similarly, for two applications to communicate, they must also agree on the specific data standards to be encoded, such as using XML; this, unfortunately, does not always happen due to conflicting specifications adopted by different vendors and agencies. A case in point is the proposed reliable-network protocol that identifies, manages, and tracks the reliable delivery of messages between source and destination WS. According to [1], there are currently two competing specifications: WS-Reliable Messaging (using a specification supported by IBM, Microsoft, BEA, and TIBCO) versus WS-Reliability (promoted by Oracle, Sun, Fujitsu, Hitachi, and Sonic Software, among others). The open source world is starting to adopt these standards feverishly in the hopes of expanding their imminent presence in the larger software market.

Case Study

The power of these open source tools and the low cost of development are precisely why Keane, a billion-dollar global business and IT consulting firm, went with a predominantly open source model for the design and development of the USAF Logistics Systems.

Through both technology and business insight, the project has been successful in its use of cost-effective interoperable open source tools to help the USAF achieve its mission and goals. The system handles 5,000 users and around 3,000 simultaneous users during peak time. The project is saving around 2 million lines of code by using a composite commercial off-the-shelf approach and utilizing an intelligent mix of open source tools. This cost-effective combination provides the USAF with the interoperability and visibility of assets across both retail and wholesale systems. It also integrates the information in near real-time into an intuitive, Web-based interface using Google Web Tools to enhance the Web 2.0 capability. The project also used an SOA-based approach to integration, leveraging the latest in open source libraries and adopting standards such as XML and the Java platform; these rapidly delivered interoperability with numerous systems and exposed legacy systems without modification.

Some core and notable open source technologies used included the Spring Framework, an application and integration framework for the Java platform; iBatis for persistence; Apache Axis, an XML-based Web service framework application development done in Java using Tomcat and extensive use of open source libraries; and Java Enterprise Edition/Java 2 Enterprise Edition components. Using these technologies to develop an innovative transformation and business rules engine is an intelligent *enterprise use* of open source technologies to achieve interoperability.

Enterprise Software Development Methodology – the Blended Agile-Rational Unified Process

For the USAF Logistics Systems, a blended approach to enterprise application development combining agile and VIEWW[2] methodologies was used to quickly deliver modernized solutions meeting 100 percent of client expectations. This collaborative development approach allowed the team to more efficiently review and test, thereby helping to speed development efforts. The Agile-Rational Unified Process (RUP) methodology is beneficial because it works with poorly understood architectures, produces a highly reliable system, produces a system with large growth capability, manages risks, can be constrained to a predefined schedule, provides management with progress visibility, and allows for in-process corrections. Figure 1 depicts the Agile-RUP methodology in some detail.

Through utilizing open source tools and innovative forward-thinking solutions, government best practices can be used as examples to follow in the commercial arena. Served industries—such as defense and aerospace, pharmaceutical, manufacturing, and financial—could benefit greatly from efficient and cost-effective enterprise-level supply chain systems, collaborative decision support engines, and intel-

ligent rules agents that change as the global economy changes.

Conclusion

I believe that the adoption of interoperable open source tools using SOA design principles will drive the future development of innovative forward-thinking solutions. Government best practices will be used as examples to follow in the commercial arena. Agencies supporting defense and intelligence—as well as the broader federal health care and financial arenas—will benefit greatly from efficient and cost-effective interoperable systems at the enterprise-level.◆

Notes

1. Learn more about BP 1.0 at WS-I: <www.ws-i.org/>.
2. VIEWW stands for the different phases of the development life cycle: Vision, Inception, Elaboration, Work, and Web. It is Keane's equivalent to the RUP.

References

1. Narang, Sanjay. "Web Service Specifications and SOA Interoperability." Enterprise Open Source Magazine. 11 Feb. 2007 <http://opensource.sys-con.com/node/314083>.

Additional Reading

1. Smith, Roger. "Using Open Source Tools to Build Interoperable Web Services." 8 Sept. 2004 <www.linux.com/feature/38651>.
2. Krafzig, Dirk, Karl Banke, and Dirk Slama. Enterprise SOA – SOA Best Practices. Indianapolis, IN: Prentice Hall PTR, 2005.
3. "Interoperability Still Stumbling Block for OpenSource in 2008." eWeek.com. 3 Jan. 2008 <www.eweek.com/c/a/Linux-and-Open-Source/Interoperability-Still-Stumbling-Block-for-Open-Source-in-2008/>.

About the Author

Shamlan Siddiqi serves as director of enterprise integration at Keane, Inc. He has more than 10 years of consulting experience working on and managing all phases of both the pre-sales and systems life cycle. At Keane, he leads the delivery of business and technical solutions to clients at all levels of the enterprise. Siddiqi has successfully led complex SOA and enterprise architecture projects in global health, the government/public sector, and for Keane's global financial services.

Keane, Inc.
1410 Spring Hill RD
STE 500
McLean, VA 22102

CALL FOR ARTICLES

If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for three areas of emphasis we are looking for:

Software Management Controls
May 2009
Submission Deadline: December 12, 2008

Modeling and Simulations
June 2009
Submission Deadline: January 16, 2009

Process Replication
July 2009
Submission Deadline: February 13, 2009

Please follow the Author Guidelines for CROSSTALK, available on the Internet at <www.stsc.hill.af.mil/crosstalk>. We accept article submissions on all software-related topics at any time, along with Letters to the Editor and BACKTALK. Also, we now provide a link to each monthly theme, giving greater detail on the types of articles we're looking for <www.stsc.hill.af.mil/crosstalk/theme.html>.