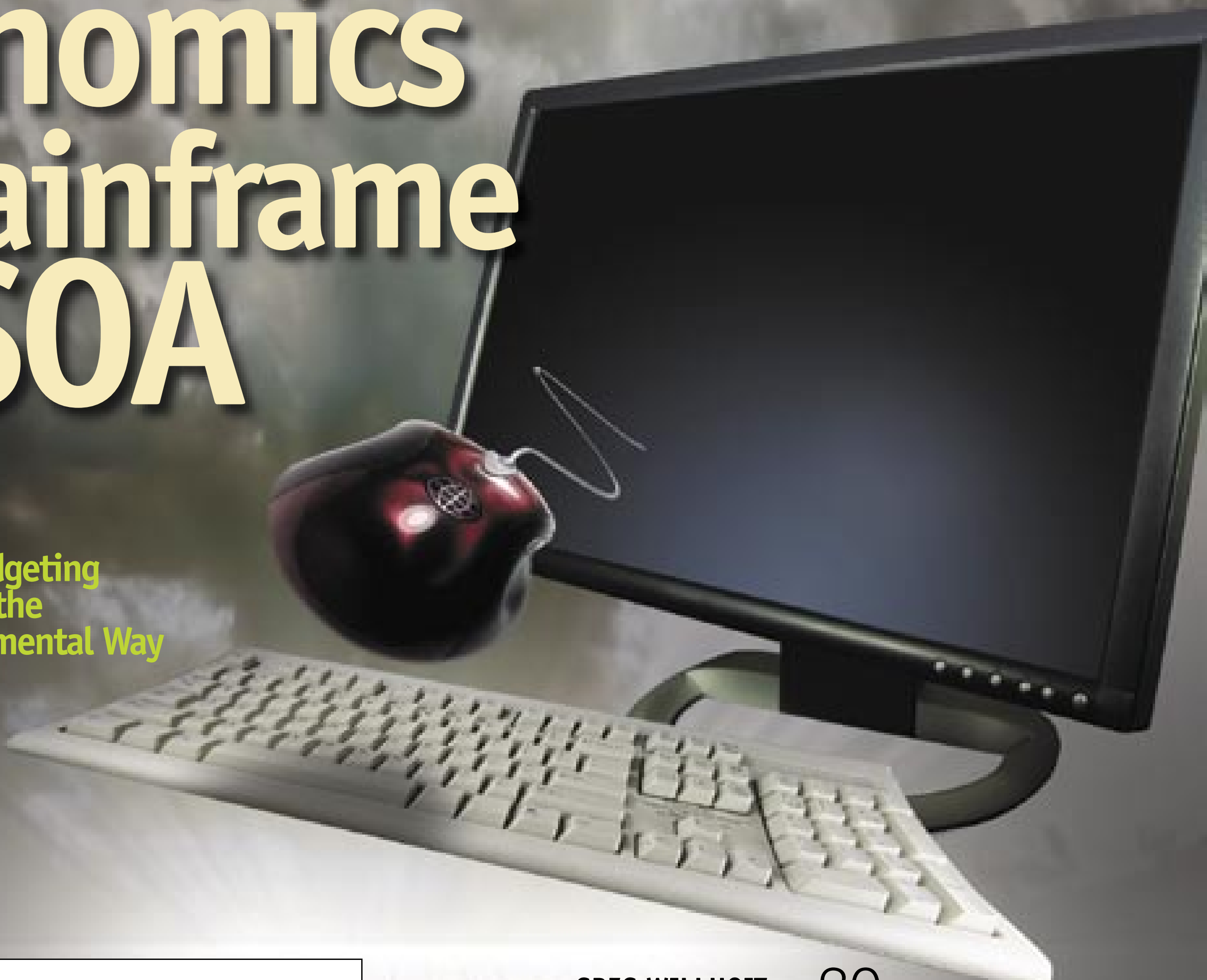
SYS-CON THE WORLD'S LEADING MAGAZINE DEDICATED TO **WEB SERVICES TECHNOLOGIES**

www.SOA.SYS-CON.com

SOA WORLD™ MAGAZINE

FEBRUARY 2008 / VOLUME: 8 ISSUE 2

Redefining the Economics of Mainframe SOA



12 **Estimating & Budgeting for SOA Projects the Iterative & Incremental Way**
NANDAN DASGUPTA

26 **The SaaS Superhighway**
ASH MASSOUDI

GREG WILLHOIT 30

JavaOne™
Sun's 2008 Worldwide Java Developer Conference℠

SOAWORLD™ MAGAZINE
www.SOA.SYS-CON.com

EDITORIAL

Editor-in-Chief
Sean Rhody sean@sys-con.com

XML Editor
Hitesh Seth

Industry Editor
Norbert Mikula norbert@sys-con.com

Product Review Editor
Brian Barbash bbarbash@sys-con.com

.NET Editor
Dave Rader davidr@fusiontech.com

Security Editor
Michael Mosher wsjsecurity@sys-con.com

Research Editor
Bahadir Karuv, Ph.D Bahadir@sys-con.com

Technical Editors
Andrew Astor andy@enterprisedb.com
David Chappell chappell@sonicsoftware.com
Anne Thomas Manes anne@manes.net
Mike Sick msick@sys-con.com
Michael Wacey mwacey@csc.com

International Technical Editor
Ajit Sagar ajitsagar@sys-con.com

Executive Editor
Nancy Valentine nancy@sys-con.com

Associate Online Editor
Lindsay Hock lindsay@sys-con.com

PRODUCTION

LEAD DESIGNER
Abraham Addo abraham@sys-con.com

ASSOCIATE ART DIRECTORS
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami @sys-con.com

EDITORIAL OFFICES
SYS-CON MEDIA
577 CHESTNUT RIDGE ROAD, WOODCLIFF LAKE, NJ 07677
TELEPHONE: 201 802-3000 FAX: 201 782-9637
SOA World Magazine Digital Edition (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending
Woodcliff Lake, NJ 07677 and additional mailing offices
POSTMASTER: Send address changes to:
SOA World Magazine, SYS-CON Publications, Inc.
577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677

SYS-CON MEDIA
www.WSJ2.com
www.SYS-CON.com

Simplicity by Design

WRITTEN BY **SEAN RHODY**

As a student of physics, Albert Einstein is one of my personal heroes. Aside from being one of the most brilliant minds to ever contemplate the universe, Albert had a way with words. One of his quotes strikes me as particularly apropos for this month's issue – "Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius – and a lot of courage – to move in the opposite direction."

This month we're focusing on two subjects that actually tie in together much more than most people actually anticipate – SOA Testing and Service Design. I'll get to how they tie together in a moment, but let's see how Einstein relates to SOA first.

With SOA, it's very easy to buy the plumbing. You go out and get an ESB, a rules engine, something to do BPEL or BPML, and something to do basic services management and you're ready to tackle that great big world of SOA.

Of course, at that point you are in the same circumstance as you were 20 years ago when you finally made that step away from ISAM and selected your first relational database. You bought all the goodies, got it installed and then you sat. And then you realized that your great big shinny database was empty. So you started designing tables. Maybe you thought about normalizing, maybe you didn't, but before you knew it you had umpteen tables and performance was in the toilet. What was so great about RDBMS anyway?

Naturally, there was a learning curve, and some bitter experience with normal forms and what level is reasonable. The question then was, and yes, I'm finally getting back to the point, what was the right approach to database design?

Fast forward to today and we're asking the same questions about service design – is it a customer service or a customer edit service? Maybe it's all part of the account service. Because after we get over learning about what WSDL is, we still have to figure out what to make with it. We still have the same issues – if we make the service too fine-grained, we have too many of them and eventually someone will come along and aggregate them anyway – and perhaps not in a way that's optimal. If we make the service too coarse-grained, we risk creating EDI all over again – one method with 80 parameters, all optional. There's even the added complexity of standards such as ACORD – importing and using the whole schema adds tons of elements, many of which you may never use. Like the man said – it's easy to make it bigger and more complex; it's hard to go the other way.

One way to try to swim against that tide is to start designing with the concept of testing in mind. Rather than wait until the service is coded before you begin to design the test cases that will check to see if it works, why not start with them. Do the design from the perspective of testability and functional completeness, and you may have a jump start on how to handle your actual service design.

There's no silver bullet that I'm aware of to magically adjust your service designs until they're optimal. Service Management provides needed tools to take a look at QoS and establish service SLAs that will allow for meaningful investigation of your service design, but the fact of the matter is you won't get it right the first time, so allow yourself time and budget for refactoring. You should also note that conditions will change over time and what was once optimized and working properly is now in need of some TLC. This is where performance testing and predictive modeling can help in anticipating capacity planning for services. Once again, an ounce of testing is worth a pound of code. You can quote me on that – I may not be Einstein, but I did stay in a Holiday Inn Select last night. ■

About the Author

Sean Rhody is the editor-in-chief of **SOA World Magazine.** *He is a respected industry expert and a consultant with a leading consulting services company. sean@sys-con.com*

INSIDE SOAWORLD MAGAZINE

DOWNLOAD PRINT VERSION

SOA Made Easy

Open Source Apache Camel

BY ROB DAVIES AND JAMES STRACHAN

Over the last several years, integration technology has been growing by leaps and bounds. The XML/REST/Web Services/SOA revolution has driven engineers and software firms to create an abundance of protocols, adaptors, transports, containers, standards, best practices...you name it.

The bits and bytes that are now available are undeniably sophisticated, diverse, and capable of almost anything, but many of the packages are built from the technology up and leave the job of how to use the capabilities effectively as an exercise for the reader.

Today, many readers have completed many such exercises. There is a wealth of experience and thousands of successful projects out there that have led to the definition of many infrastructure design patterns that help developers cut to the chase when it comes to integration. One set of design patterns that has gained traction in the industry is Hohpe and Woolf's Enterprise Integration Patterns. These patterns include a technology-agnostic vocabulary for describing large-scale integration solutions. Rather than focusing on the low-level programming, they take a top-down approach to developing an asynchronous, message-based architecture.

A consistent vocabulary is nice, but an easy-to-use framework for actually building the infrastructure would be even better.

That was exactly the thinking behind the open source Camel project at Apache. Now that a tried-and-true set of patterns is available, the obvious next step is to create an engine that can implement the patterns in the simplest way possible.

Camel is a code-first tool that allows developers to perform sophisticated large-scale integration without having to learn any vendor-specific or complex underlying technology. Camel is a POJO-based implementation of the Enterprise Integration Patterns using a declarative Java Domain Specific Language to connect to messaging systems and configure routing and mediation rules. The result is a framework that lets Java developers design and build a Service Oriented Architecture (SOA) without having to read pages and pages of specifications for technologies like JMS or JBI or deal with the lower-level details of Spring.

Apache Camel grew organically from code and ideas that were generated from other Apache projects particularly Apache ActiveMQ and Apache ServiceMix. Project members found that people wanted to create and use patterns from the Enterprise Integration Patterns book in many different scenarios. The Camel team set about to build such a framework for exactly this purpose.

Camel Overview

The first step in building Camel was to decouple the implementation of the patterns from the underlying plumbing. Some people want to use the patterns inside an enterprise service bus (ESB), some people want to use them inside a message broker, and other people want to use these patterns inside an application itself or to talk between messaging

SOAWORLD MAGAZINE
www.SOA.SYS-CON.com

CORPORATE
President and CEO
Fuat Kircaali fuat@sys-con.com
Senior VP, Editorial & Events
Jeremy Geelan jeremy@sys-con.com

ADVERTISING
Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com
Advertising Sales Director
Megan Mussa megan@sys-con.com
Associate Sales Manager
Corinna Melcon corinna@sys-con.com
Advertising Events Associate
Alison Fitzgibbons alison@sys-con.com

SYS-CON EVENTS
Event Planner
Sharmonique Shade sharmonique@sys-con.com

CUSTOMER RELATIONS
Circulation Service Coordinators
Edna Earle Russell edna@sys-con.com

SYS-CON.COM
Consultant information systems
Robert Diamond robert@sys-con.com
Web Designers
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

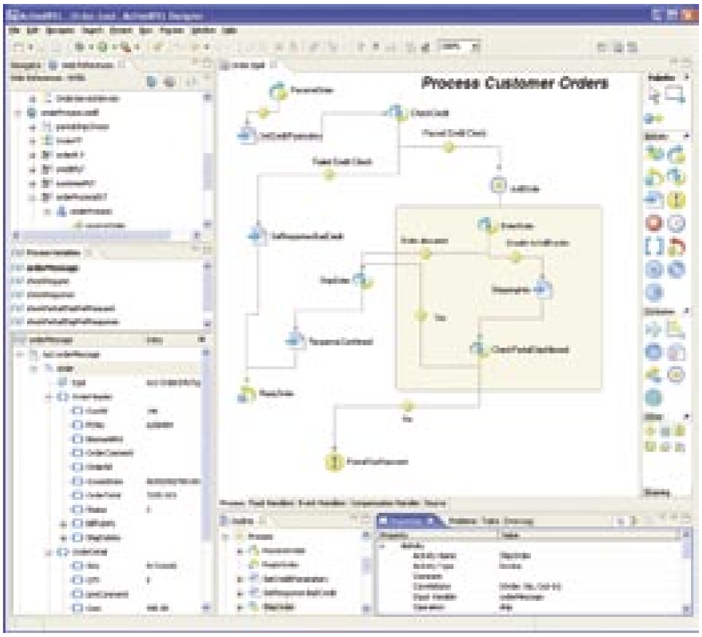
ACCOUNTING
Financial Analyst
Joan LaRose joan@sys-con.com
Accounts Payable
Betty White betty@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1-201-802-3012 or 1-888-303-5282
For subscriptions and requests for bulk orders,
please send your letters to Subscription Department
Cover Price: $6.99/issue
Domestic: $99/yr (12 issues)
(U.S. Banks or Money Orders)

For list rental information:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com;
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.

SYS-CON MEDIA www.SYS-CON.com


Process Customer Orders

providers. Still other people want to use them inside a Web Services framework or some other communication platform. Rather than tie this routing code to a particular message broker or ESB, Camel extracts this code to be a standalone framework that can be used in any project. Camel has a small footprint and can be reused anywhere, whether in a servlet, in the Web Services stack, inside a full ESB, or in a messaging application.

The primary advantage of Camel is that the development team doesn't have to work with containers just to connect systems. Many might consider working with containers to be a right of passage or a test of one's mettle, but to a growing number of teams these hurdles are an unnecessary barrier to entry. With Apache Camel, developers can get the job done with a minimum of extraneous tasks. Camel can, however, be deployed within a JBI container if other requirements warrant that, but it's not necessary.

To simplify the programming, Camel supports a domain-specific language in both Java and XML for the Enterprise Integration Patterns to be used in any Java IDE or from within spring XML (see Figure 1). This higher level of abstraction makes problem solving more efficient.

Figure 1: Supported Enterprise Integration Patterns

Camel reuses many Spring 2 features, such as declarative transactions, inversion of control configuration, and various utility classes for working with such things as JMS and JDBC and Java Persistence API (JPA). This raises the abstraction level to make things very simple, reducing the amount of XML one has to write, but still exposing the wire-level access if anyone needs to roll his sleeves up and get down and dirty.

Camel Examples

We're going explain different ways of configuring Apache Camel, first using the Java DSL (Domain Specific Language) and then using Spring XML configuration.

Java DSL Configuration

This example demonstrates a use case in which you want to archive messages from a JMS Queue into files in a directory structure. The first thing to do is to create a CamelContext object:

```
CamelContext context = new DefaultCamelContext();
```

There's more than one way of adding a Component to the CamelContext. You can add components implicitly – when we set up the routing – as we do here for the FileComponent:

```
context.addRoutes(new RouteBuilder() {

    public void configure() {
        from(“test-jms:queue:test.queue”).to(“file://test”);
        // set up a listener on the file component
        from(“file://test”).process(new Processor() {

            public void process(Exchange e) {
                System.out.println(“Received exchange: “ + e.getIn());
            }
        });
    }
});
```

or explicitly – as we do here when we add the JMS Component:

```
ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(“vm://
localhost?broker.persistent=false”);
// note we can explicity name the component
context.addComponent(“test-jms”, JmsComponent.jmsComponentAutoAcknowledge(connect
ionFactory));
```

Next you must start the Camel context. If you're using Spring to configure the Camel context this is done automatically for you; although if you're using a pure Java approach then you just need to call the start() method:

```
camelContext.start();
```

This will start all of the configured routing rules.

So after starting the CamelContext, we can fire some objects into Camel.

In normal use, an external system would be firing messages or events directly into Camel through one if its components but we're going to use the CamelTemplate, which is a really easy way to test your configuration:

```
   CamelTemplate template = new CamelTemplate(context);
```

We can now send some test messages over JMS using the CamelTemplate:

```
for (int i = 0; i < 10; i++) {
    template.sendBody(“test-jms:queue:test.queue”, “Test Message: “ + i);
}
```

From the CamelTemplate we send objects (in this case text) into the CamelContext to the Component test-jms:queue:test.queue. These text objects will be converted automatically into JMS Messages and posted to a JMS queue named test.queue. When we set up the route, we configured the FileComponent to listen of the test.queue.

The file FileComponent will take messages from the queue and save them to a directory named test. Every message will be saved in a file that corresponds to its destination and message ID.

Finally, we configured our own listener in the route to take notifications from the FileComponent and print them out as text.

Spring XML Configuration

This example will use Spring XML configuration to transform files from a directory using XQuery and send the results to a JMS queue. It parsers some files from a directory, transforms them using XQuery then sends them to a message queue. To make it easy to look at the generated files, we also have another route that consumes from the JMS queue and writes them to an output directory.

Running the Example

To run the example we use the Camel Maven Plugin. For example, from the source or binary distribution the following should work:

```
cd examples/camel-example-spring-xquery
mvn camel:run
```

You should now see the generated files in the target/outputFiles directory, which are the transformed messages read from the JMS queue.

Code Walkthrough

What this does is boot up the Spring ApplicationContext defined in the file META-INF/spring/camelContext.xml on the classpath. This is a regular Spring XML document that uses the Camel XML configuration to configure a CamelContext.

Note that at the end of this XML example file we explicitly configure the ActiveMQ component with details on how to connect to the broker.

The main part of the Spring XML file is here:

```
<camelContext useJmx=”true” xmlns=”http://activemq.apache.org/camel/schema/
spring”>

  <!-- lets parse files, transform them with XQuery and send them to JMS -->
  <route>
    <from uri=”file:src/data?noop=true”/>
    <to uri=”xquery:myTransform.xquery”/>
    <to uri=”jms:MyQueue”/>
  </route>

  <!-- now lets write messages from the queue to a directory -->
  <route>
    <from uri=”jms:MyQueue”/>
    <to uri=”file:target/outputFiles”/>
  </route>

</camelContext>

<!-- lets configure the default ActiveMQ broker URL -->
<bean id=”jms” class=”org.apache.camel.component.jms.JmsComponent”>
  <property name=”connectionFactory”>
    <bean class=”org.apache.activemq.ActiveMQConnectionFactory”>
      <property name=”brokerURL” value=”vm://localhost?broker.persistent=false”/>
    </bean>
  </property>
</bean>
```

This hopefully has given you a flavor of how easy it is to do enterprise integration using Apache Camel. For more information see the Web site at http://activemq.apache.org/camel/.

Resources

http://www.enterpriseintegrationpatterns.com/ ■

About the Authors

Rob Davies, director of open source development at IONA, has more than 20 years of experience developing high-performance distributed enterprise systems and products for telecom and finance corporations. He is responsible for leading the development of IONA's FUSE family of open source products, which are based on leading projects at the Apache Software Foundation. Rob is a founder of the Apache ActiveMQ, Apache ServiceMix and Apache Camel projects. Prior to joining IONA, Rob served as the founder and vice president of product development at LogicBlaze, which was acquired by IONA in 2007. Previously, Rob served as founder and CTO of integration software developer SpiritSoft.

James Strachan, technical director at IONA, is responsible for helping the Company provide open source offerings for organizations requiring secure, high-performance distributed systems and integration solutions. He is heavily involved in the open source community, and has co-founded several Apache projects, including ActiveMQ, Camel, Geronimo and ServiceMix. He also created the "Groovy" scripting language and additional open source projects such as dom4j, jaxen and Jelly. Prior to joining IONA, James spent more than 20 years in enterprise software development. Previously, James co-founded LogicBlaze, Inc., an enterprise open source company acquired by IONA. Prior to that, he founded SpiritSoft, Inc., a company providing enterprise Java middleware services.

Advertiser Index

ADVERTISER	URL	PHONE	PG
Active Endpoints	www.active-endpoints.com/soa		7
AJAXWorld Conference & Expo	www.ajaxworld.com	201-802-3021	27
AJAXWorld Conference & Expo	www.ajaxworld.com	201-802-3021	21
Altova	www.altova.com		36
Attachmate	www.attachmate.com/soa		15
Fiorano	www.fiorano.com/downloads		5
SOA World Conference & Expo	http://soaworld2008.com/		17
JavaOne 2008	http://java.sun.com/javaone		2
SYS-CON Books	http://www.books.sys-con.com/	201-802-3020	25
SYS-CON	www.sys-con.com	201-802-3020	33
WSO2	http://wso2.com/products/esb/		35

Sanity, Strategy, Sustainability

The new world of verification platforms

BY MARK UNDERSETH

The lines between embedded and enterprise computing are blurring. As embedded devices become more and more networked, they form a perimeter at the edge of our enterprise data centers and a rich source of real-time, real-world information.

These newcomers to the enterprise are coming of age. Embedded computing is moving beyond the world of hidden computing and a behind-the-scenes role in military and industrial applications. Embedded applications are no longer bench projects hacked together by a few elite engineers. Embedded computing now comprises mainstream applications that are a part of our daily lives and it is big business.

As the embedded computing industry evolves to support an ever-growing range of capabilities and features, embedded applications are becoming increasingly complex. Take, for example, our cell phones. They not only make calls, but send and receive text messages, surf the Internet, take photographs, and play music. While these devices pack more hardware into a given space to enable these features, they are fundamentally driven and operated by software. We've gone from a few thousand lines of code in mobile phones 10 years ago to millions of lines of code today, running on full-fledged, 32-bit, multitasking embedded operating systems.

With the software content in electronic products doubling every two years, manufacturers are finding they can no longer do everything themselves. To keep up, companies employ more engineering resources across the globe. Today, embedded-software development involves a global supply chain of engineers, offshore outsourcers, open source and third-party software vendors. All of them are delivering hundreds of software components (see Figure 1).

Despite increases in software complexity, engineering processes and techniques are essentially the same as they were 10 years ago. While the emerging global supply chain for embedded software has helped in creating new code components, it has made it much more difficult to integrate them. This is because software test and verification methods that worked well enough when embedded projects were smaller and simpler don't scale sufficiently for today's bigger and more complex efforts. As a result product teams encounter a frustrating and all too familiar scenario. Software development generally appears to progress smoothly through the implementation phase. However, from that point on, projects enter long, churning, and unpredictable integration and system test phases. Companies struggle to integrate and verify software from so many sources. As a short-term fix, software managers add more engineers and resources to do the integration and testing. But this approach is limited and expensive. Software releases are still delivered late with compromised quality. Clearly, just adding more people alone won't solve the problem.

Enterprise and desktop computing faced similar challenges as they evolved. To a large degree, both have successfully managed software complexity by moving toward standardized application platforms such as Windows or Linux running on an Intel processor. Embedded computing is more complex, however, because applications must run on different processor families and in different software environments. The challenge of testing and verifying a huge array of software components on all this hardware can be overwhelming. Without achieving a scalable, reliable, and predictable software process, software verification and integration challenges will only worsen with each new generation of embedded hardware.

In the absence of a standardized embedded application platform, the need for a sustainable verification platform becomes compelling. A sustainable verification platform would provide a scaleable shared framework that could be used across the product teams at all stages of the release cycle. It could be reused on next-generation projects. It could even be extended across organizations, vendors, and suppliers across the globe. The question is how do we achieve this? Neither in-house efforts nor outsourcing have fully addressed the problem. New automated processes, innovative technology, and specialized verification expertise are required. A new business solution, verification-as-services aims to fulfill this need and enable smoother integration, more predictable schedules, and better quality products.

A Fresh Approach To Verifying the Software Supply Chain

A new model is required to significantly optimize the delivery of highly integrated complex embedded software. To address the urgent challenges observed in the integration and system test phases, key objectives of the new model would be to validate software components earlier; reuse and integrate code more effectively; automate more test processes, and increase visibility into software quality earlier in the development cycle (see Figure 2).

The centerpiece of this new approach is the so-called test assets. Test assets are fully automated reusable tests that have lasting value because they can be leveraged throughout the development lifecycle, in derivative projects, and by other functional teams to integrate, triage, and verify software. Test assets also provide detailed and ongoing visibility into the health of the software. In an ideal world, and according to the principles of supply chain management, all software suppliers, both internal and external, would create and deliver test assets with their software.

To create and share test assets effectively a unified verification framework is required. The verification framework provides the common tools and environment needed for collaboration with suppliers, better leverage of domain expertise, and highly automated testing and early verification strategies. It provides the common infrastructure to reuse, automate, and execute tests for all components effectively, and to support test management and reporting. The verification framework must facilitate different kinds of testing such as unit level and API-level testing. Further, it must scale well beyond a single engineering group and be able to aggregate test assets from both internal engineering teams as well as external software suppliers.

Besides the verification framework and test assets, new processes must also make the best use of new capabilities. New processes will leverage, reuse, and apply automated test assets as much as possible to verify software at various stages and levels – by individual developers, by functional teams, on fully integrated engineering builds, daily, hourly, even continuously. With automated reporting, the processes would also provide comprehensive quantitative metrics and visibility into quality and completeness at various stages of development (see Figure 3).

Developing a Sustainable Verification Platform

Virtually all embedded software development groups recognize that software verification is a problem but lack an effective and ongoing means of addressing it. Development teams aren't really able to solve the problem effectively because they're trapped in a vicious cycle: they're too busy developing and fixing problems and have no time to do the heavy lifting required to adopt the technology required to implement a verification strategy. Traditional outsourcing efforts often fall short because they fail to provide the specialized capabilities and the scaleable and reusable infrastructure required for embedded application development.

The verification-as-services (VaS) approach brings together the verification expertise, innovative technology, and ongoing process management needed to provide a sustainable platform for verifying embedded code. Verification-as-services strives to leverage the strengths of the software-as-a-service (SaaS) model popularized by salesforce.com while applying a managed services model for the infrastructure and technology designed specifically for embedded software verification. VaS means an organization doesn't have to manage a large staff of test and verification engineers, because the resources and domain expertise needed are provided as part of the service. The VaS business model focuses on value delivered. Unlike outsourcing models that charge according to headcount, VaS is a variable cost solution that is priced according to the number of components to be verified, the scope of testing required, and the specific goals of the project.

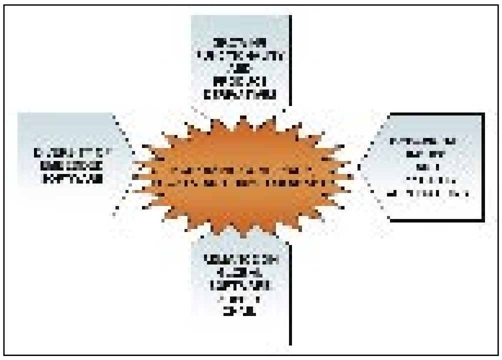


Figure 1:

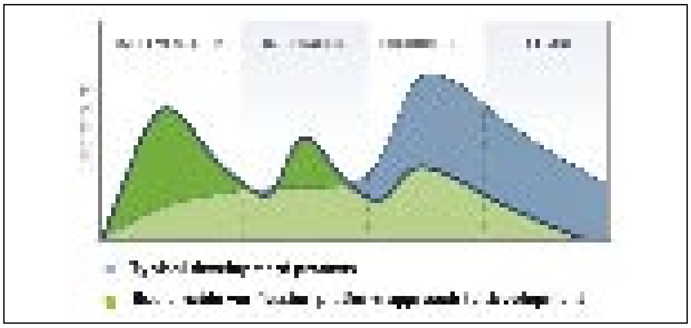


Figure 2:

Figure 3:

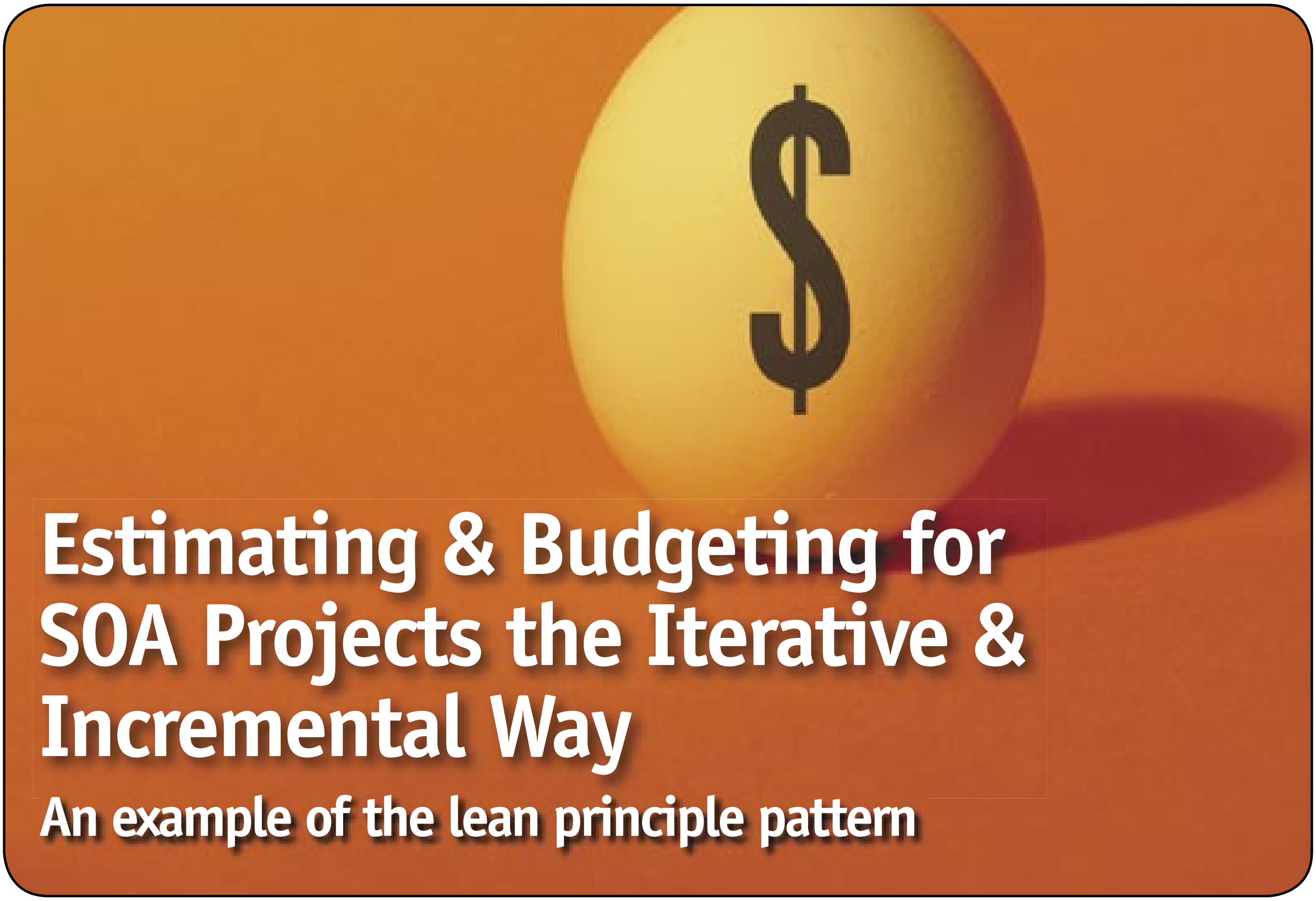
With VaS, all verification activities are codified as part of a monthly subscription service that provides complete management of embedded software verification. This approach eliminates large upfront software license costs and ensures an organization continues to get full value from the solution over time.

A Sustainable Verification Platform in Action

To implement a verification strategy that meets an organization's unique software challenges, corporate culture, and project timelines, testing and integration experts first interview, observe, and document existing development processes. Next the verification platform is integrated into the key systems used in those processes. Finally, verification experts set up software, processes, and reports and execute verification activities as an ongoing service for development groups.

VaS incorporates a scalable infrastructure that removes the obstacles for early test and validation success. At every stage, development teams will be able to seamlessly blend testing and verification activities, including planning, analysis, test execution, and measurement, with the organization's embedded software development processes. Because most development groups have other high-priority objectives to accomplish, VaS automates testing processes, executes tests, certifies components according to predetermined criteria and applies product enhancements on an ongoing basis during the life of the subscription.

– CONTINUED ON PAGE 20



Estimating & Budgeting for SOA Projects the Iterative & Incremental Way

An example of the lean principle pattern

BY NANDAN DASGUPTA

"Ethical axioms are found and tested not very differently from the axioms of science.

Truth is what stands the test of experience." – Albert Einstein

From the mainframe to the Internet age software projects have gotten delivered or deployed in increments. A whole project is never delivered or deployed on a given date.

The iterative and incremental way of managing projects is popular because the customer pays as the deployment is done. So a project manager has to estimate and budget for each increment and give management regular updates of the time and cost remaining and research the reason for any cost deviations.

SOA will take IT to a new agile state. It's not a development roadmap but a functional way of executing business processes with various IT avenues. It mainly means moving from multiple applications to a shared service-based environment.

A key aspect of SOA is the definition of roles and their importance. A business process architect will set the direction, the architecture role will define the keys, system process architects define the detail paths, and other roles will follow the flow.

Lean is a process method that lays a timing framework and quantity factor or a framework of schedules. The term "lean manufacturing" is embodied in the "Toyota Production System," where it was first applied and follows a philosophy combining tools, techniques, and resources.

The core concept is to increase value and decrease waste in a process/project/organization. Lean is a method that lays a process framework for delivering a finished product with maximum time-value and benefit realization.

SOA is an incremental and iterative process that through an incremental development and periodic review will achieve the desired result. So a project manager has to iteratively plan increments and iteratively estimate and budget for each increment. This process can be enhanced by following "lean principles."

Thoughts on Effective Estimating and Budgeting

Estimation is defined as quantitative assessment of the cost of all resources to complete a project or an initiative. Budgeting is defined as a plan for obtaining the necessary financial resources for implementing a project or an initiative.

In any organization the core future plan is emphasis on measurement of value of financial aspect. Financial planning is iterative but setting the capital budget is a one-time event taken at the beginning of year. First there's the go and no-go decision then the budget for the release/increments (the project milestones) is calculated. A cost monitoring and cost variance analysis is done to improve the next increment/release and assess the benefits in terms of NPV, IRR, and payback period.

Footsteps of Pattern for Iterative and Incremental Development

Managing project activities produces results in small chunks and accrue early and continuous benefits.

- *Iterative: Repeat essentially the same process.*
- *Iteration Planning: The project team decides what to do in which order.*
- *Release/Incremental: Deliver usable functionality in chunks.*
- *Release/Increment Planning: Customer/stakeholder selects the scope of a release increment and subsequent releases/increments.*

Such a process provides opportunities to adjust the project due to changes in requirements, risks, and other items. It facilitates incorporating changes in the next iteration/release/increment. The duration of an iteration/release/increment depends on what has to be done. If a deadline isn't met then some tasks can be moved to a later release(s)/increment(s).

Per cost planning calculates the total cost of each release/increment and the total direct costs (activity-independent) of each release/increment. So as a release/increment progresses one assesses latent capacity and scalable capacity, the availability of the current architectural configuration, and the likely effects of architectural modifications (see Figure 1). When an increment/iteration is done, an "assessment" is made on how to adjust and improve the next/future increment(s)/release(s).

Using Lean Principles

Lean principles in software development define a practical framework for deploying increments or releases. They focus on reducing the "Seven Wastes" [www.leansoftwaredevelopment.org]:

- Overproduction
- Waiting time
- Transportation
- Processing
- Inventory
- Motion
- Scrap in manufactured products or any type of business

The lean method in software development is the iterative and incremental development of a solution that advocates that each increment/release be responsible for all activities, skill knowledge, and iteratively work to its release goal (see Figure 2).

Directions for Using SOA

SOA facilitates a set of services. A service is a standalone business function or process.

In SOA, services cover online interfaces and batch interfaces. Orchestration is the nerve center that directs the processes, the next command line calls different routines/batch cycles/interfaces then there are functional executables that execute or trigger a data/infrastructure services and return data/information to the main calling center. The point in structuring services is associating a number of functions/operations to the service. So one should associate more functions/operations to a service than minimally possible and less than maximally possible.

The strategy in developing SOA is we need a "process" and "architecture plan." The administrative part has to deal with the determination of business requirements, what services to constitute, what services to reuse, and what application it will operate.

Case Study

Let's consider a SOA case study of a "material management system" of an airline manufacturing company.

Objective

Rapid company growth has caused business components to grow. So there is an increased growth in inventory and subsequent stock updating, purchasing, billing and accounting activities.

Challenge

The airplane manufacturer makes "large commercial airplane" modules at several localized units and at other company units around the globe. So several "software applications" were developed and each of them supported different business processes and have their own data repositories. So at a given point of time the company has no consolidated information about its Inventory, stock-in-hand, purchasing of item numbers, exercising billing figures, or accounting activities (see Figure 3).

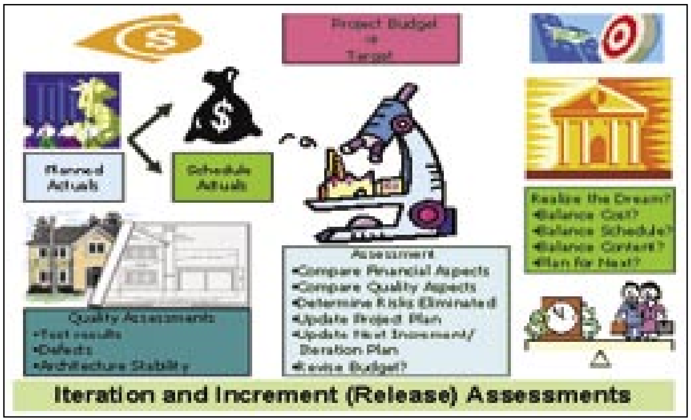


Figure 1:

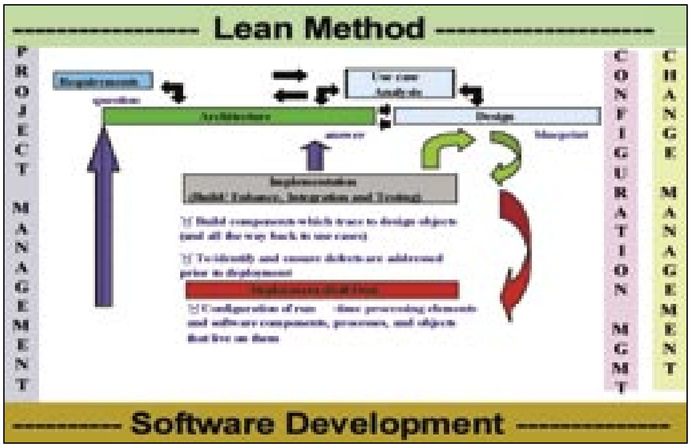

Lean Method
PROJECT MANAGEMENT
Requirements
question
Use case
Analysis
Architecture
Design
answer
blueprint
Implementation
(Build, Enhance, Integration and Testing)
Build components which trace to design objects
(and all the way back to use cases)
To identify and ensure defects are addressed
prior to deployment
Deployment (Roll Out)
Configuration of run-time processing elements
and software components, processes, and objects
that live on them
CONFIGURATION MGMT
CHANGE MANAGEMENT
Software Development

Figure 2:

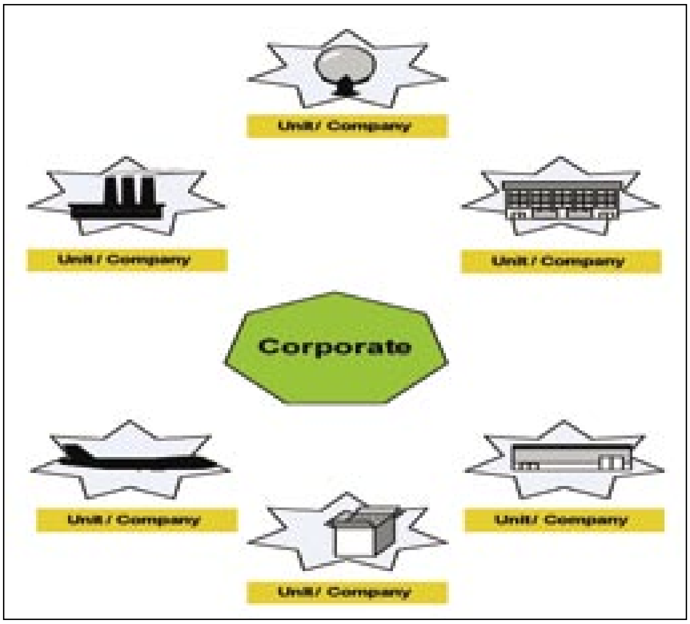


Figure 3:

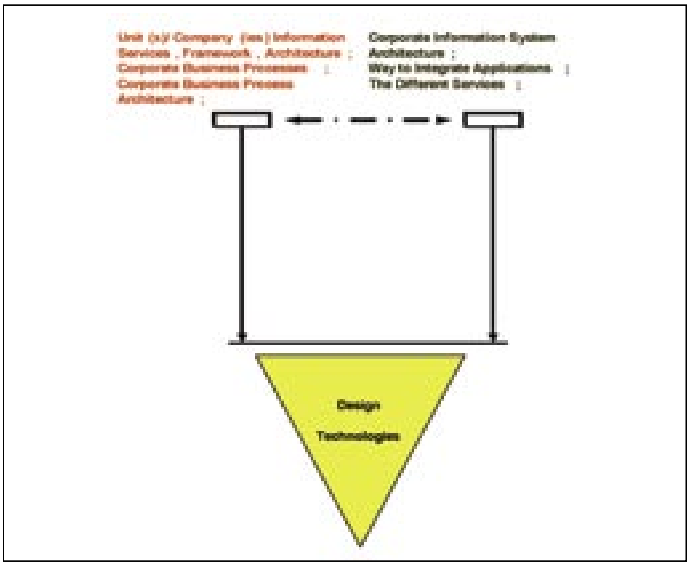

Unit (s)/ Company (ies) Information Services , Framework , Architecture ;
Corporate Business Processes ;
Corporate Business Process Architecture ;
Corporate Information System Architecture ;
Way to Integrate Applications ;
The Different Services ;
Design
Technologies

Figure 4:

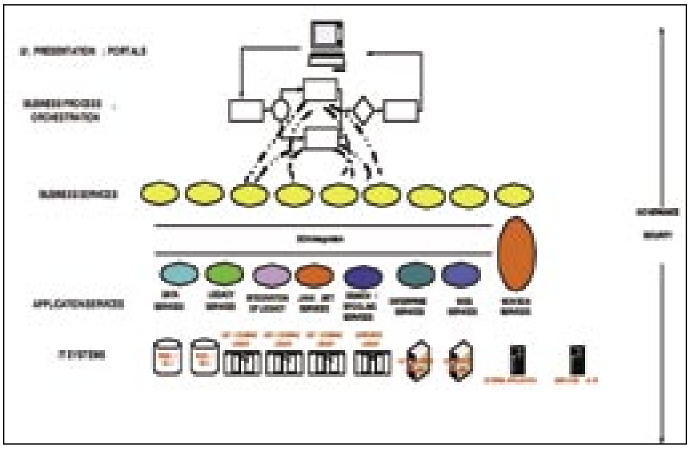


Figure 5:

Solution

Let's use SOA principles in solving this organization problem. SOA can be mixed and matched to any business process significantly reducing development costs. Meaning SOA isn't a development roadmap but a functional way to execute business processes with various IT avenues. SOA promotes a new "multiple use," not only reuse. So it promotes a simple process of pointing to a service and accessing it. The prime aspects of SOA in an organization/ enterprise are:

- Process orientation
- Adaptable to change
- Deals with different technologies
- Deploys across systems

SOA radically changes the idea of operating in multiple applications to shared service-based environments (see Figure 4).

So in constructing a SOA project in an enterprise it must address:

- **Strategies for the Project:**
  - What are the project benefits? – They will determine what to build
  - What is the collaboration between various parties (management, business, and IT) to facilitate idea sharing?
  - What team members will strategize and execute the SOA?
  - What are the functional cases that will drive SOA?
  - What are the detailed flows between various disciplines ranging from requirements to deployment?
  - What is the orchestration that directs the processes?
  - What are the crucial assets of SOA, the "services"?
  - What are the different services that can perform for use in legacy, Web Services, and without Web Services?
  - How to integrate applications? Use an ESB?
  - What different parallel directions can be etched across the Enterprise Resource Layer(s) (data sources)?
  - What are the different technologies to use?
  - What will be the challenges in managing SOA?
  - What are the real benefits of using SOA?

- **Issues of Project Scheduling:**
  - What is the incremental/releases pattern of the project?
  - What is the duration of each increment/release?
  - What functionality (Use cases) is planned in an increment/release?
  - What are the solutions for the various risk factors?
- What if the end date of an increment/release isn't met?
- What happens when the technology associated with a function becomes obsolete?
- What happens when key personnel changes?
- What happens if there are changes in the corporate strategy?

- **Issues in estimating and budgeting the project:**
  - What is the cost estimate for identifying significant architectural requirements and the outlining architecture?
  - What are the cost estimate for build/buy/reuse (internal)/use (external facility) decisions for a part of vision or for the entire vision?
  - What is the cost estimate for doing a project plan or increment/release plan?
  - What is the project's budget?
  - What are the benefits elements, NPV, IRR, ROI?

Making decisions with only half the information is ... useless.

SOA is a moot point if you don’t include your mainframe systems

There are a lot of vendors claiming they can solve integration issues, but very few have the know-how to actually get to the heart of your business — your legacy system. And if you can’t get all the information, your plans for smooth integration and a service-oriented architecture come to a standstill.

Our time-proven technology lets you SOA-enable your host, so you can respond to new business demands economically, by maximizing the investments you’ve made in your legacy assets over the years. Without taking unnecessary risks.

Find out how AT&T, Freightliner, PPG Industries and other leading companies have quickly and cost effectively incorporated their legacy systems into Service Oriented Architectures. Download the white paper at www.attachmate.com/soa.

attachmate®

Strategy for a SOA Framework

Step One: Direction

So the organization won't go for writing software modules. It wants to build services integrating information from service providers with a bus and deliver information to various service consumers (see Figure 5).

Step Two: Presentation

Frame a customer portal/Web site that will offer resources of inquiry, update, and search engines.
Technology: STRUTS, TILES, JFACES, and AJAX.

Step Three: Business Components

Identify the business components of the system and document the system's functional requirements – Stock inventory, invoices, accounting, costing, billing, purchase, and HR.
Technology: RequisitePro, DOORS, and UML Modeler (IBM Rational Rose and ClearCase).

Step Four: Service Orchestration

The first step is determining the services and process flows in the existing systems. The second step is to lay down the flow of the existing services and new services that matches the business solution. The third step is etching out a process model of the new system.
Deliverables and Technology: BPEL, WSDL, XSD, XML, and UDDI.

Step Five: Business Services

Identify reusable business services – Stock update, year-before-last update, purchase requisition, GL accounts update, purchase master update, billing requests, and employee validation.
Technology: XML, WSDL, XML, XSLT, SOAP, UDDI, MQSeries, and JDBC.

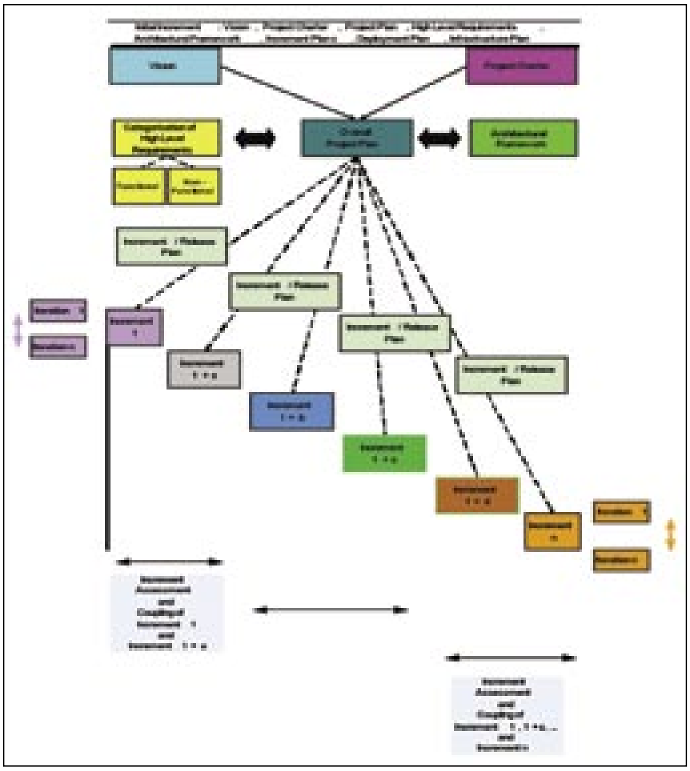


Figure 6:

Step Six: Integration Layer/Bus

Identify the integration layer.
Technology: WebSphere, WebLogic, and CORBA.

Step Seven: Application Services

Identify the service repository – data services, legacy services, the integration of legacy services, search/ spooling services, Web Services, and others.

Deliverables and Technology: SQL, ETL, adapters, XML, XMI, SOAP, UDDI, MQSeries, CICS transaction gateway, IMS Connect, ApplinX, Winsurf mainframe access, EntireX, DPL, 3270 Bridge, JDBC, XLinker, Artix, FUSE, SQL, and PL/SQL.

Step Eight: Non-SOA Services

Identify any non-reusable services or standalone services of the system.

Deliverables and Technology: SQL, ETL, adapters, XML, XMI, SOAP, UDDI, MQSeries, CICS transaction gateway, IMS Connect, ApplinX, Winsurf mainframe access, EntireX, DPL, 3270 Bridge, JDBC, XLinker, Artix, FUSE, SQL, and PL/SQL.

Step Nine: IT Systems

Identify the IT systems – Databases, legacy applications, servers, external applications, employee and HR systems.
Technology: IBM 3090, AS/400, VSAM, IMS, DB2, RDBMS, JAVA, .NET, SAP, PeopleSoft, and COBOL.

Step 10: Governance (configuration management, change management) and Security

Frame the draft on the governance (configuration management, change management) and security aspects.
Technology: Tivoli Access Manager, ClearCase, and Serena.

Incremental Project Scheduling – The Lean Method

SOA is an incremental and iterative process that through periodic incremental development and review will achieve the target. The main recommended directions that control the incremental and iterative development of SOA are:

- **Overall Project-Level Stage**
  - List the high-level building blocks of the system
  - List the components of each building block
  - Assess the different technologies
  - Define the components' access points
  - Work out the collaboration between the components to provide services
- **Increment/Release Project-Level Stage of functional case(s) that will drive this increment/release**

- List the components in this subsystem
- Determine the elements of the components
- Determine the number and range of the business processes
- Validate the activities of the internal and external services
- Determine the number and range of the business services
- Determine the number and range of the application services
- Determine the number and range on non-SOA services
- Estimate the data needs and figure out the average projected impact
- Assess the integration of the different technologies
- Reckon the security challenges

The project team would build the whole program in increments and the deliverables of each increment of a workable or executable system (a subsystem of the whole system). In each increment the team would be composed with players with the required skill sets and they would be responsible for deploying the module, maintaining the module after deployment and adding the module to other modules. The team would follow the "Lean Method" and the goal is to reduce waste and generate more value in the deliverable(s).

Each increment will be given human resources with required skill sets (business analysts, data analysts, data modelers, architects/developers, developers/testers) and they are totally responsible for the increment deliverable. So resources working on one increment can be assigned to other increments.

There will be one "initial increment" cycle in which the vision, project charter, project plan, high-level requirements, architectural framework, increment plans, deployment plan, and infrastructure plan are planned and documented.

- A business team will document the high-level business requirements and prioritized them by their importance
- A technical team lays the architectural framework and assesses the architectural risks of each component of the layout. Then the team will document high level non-functional requirements

For the SOA case study of a material management system for an airplane manufacturing company, the project manager will plan and prioritize the increments by functionality and by framework.

The work breakdown structure is:

- Functionality issues will be covered in "Increment 1" through Iterations (1 to n)
- Functionality receipts will be covered in "Increment 1 + a" through Iterations (1 to n)
- Assess "Increment 1" and "Increment 1 + a"
- Rework "Increment 1" and "Increment 1 + a" systems through Iterations (1 to n)
- Verify with business that the features of "Increment 1" and "Increment 1 + a" are working with test data and production data
- Couple "Increment 1" and "Increment 1 + a" through Iterations (1 to n)
- Functionality Stock Update Master will be covered in "Increment 1 + b" through Iterations (1 to n)
- Functionality Year-Before-Last Update will be covered in "Increment 1 + c" through Iterations (1 to n)
- Assess "Increment 1 + b" and "Increment 1 + c"
- Rework "Increment 1 + b" and "Increment 1 + c" systems through Iterations (1 to n)
- Verify with business that the features in "Increment 1 + b" and "Increment 1 + c" are working with test data and production data
- Couple "Increment 1," "Increment 1 + a," "Increment 1 + b," and "Increment 1 + c" through Iterations (1 to n)

So if all the features don't make an increment's delivery date we would close the increment/release on the date assigned with the build and features tested. Then another increment would start with the same team adding the leftover features (Increment 1 would begin with Increment E1).

Similarly if the technology or key personnel change, the project manager/team would revisit the parameters and refine the tools and resources for the increment. Finally if the corporate strategy dictates some different direction the project manager/team would tabulate the features already build and submit them to the project office for direction.

So the primary goal is delivering and deploying the working part(s) of the system. The secondary goal is adding each working part to deliver the whole system.

So the project manager/team follows the Lean Method by adding value to the business and delivering the working parts of the whole system and we improve the strategy of future increments by learning from the previous increments (see Figure 6).

Estimating & Budgeting for Increments & Iterations Using a Lean Pattern

Corporate finance allocates funds to a project by assessing cost estimate, benefits (forecast) in terms of financial metrics (ROI, NPV, IRR, and payback period). The main criteria it reviews are the expenditure amounts and timing and the income amounts and timing.

So at the beginning of any project one of the first challenges is to estimate the resources needed and work up a draft budget. To meet the criteria given above the project manager has to identify the constituent tasks, work out the resources, and calculate the funding required by the resources.

An incremental approach estimate balances the incremental releases against the constraints imposed by the overall project framework.

An iterative approach estimate reconciles the resources and funding required by a constituent task with a forecast of needing more resources. This later forecast considers the different growth scenarios and calculates the additional resources (the difference between the "to-be" resources and the "as-is" resources).

Iterative and incremental budgeting is a corporate governance tool whose function governs operational costs by cost planning and control.

"The Lean Principle means adding value to the business by delivering working parts of the whole system incrementally and executing within allocated funds"

- The aims of cost planning are cost forecasting and calculating cost objectives.
- The goal of cost control is to define the overall project funding and compare and audit how each incremental release spent the funds.

Incremental budgeting is to gather funds for an increment/release and audit how the funding is used.

Iterative budgeting (of an increment) is done to know what each iteration will cost and combine the funding of each iteration so that it's within the increment budget's limits.

For the SOA case study of the material management system for an Airplane maker, the project manager will iteratively execute the estimates and budgets for each increment.

So in estimating and budgeting for the initial increment the first step is to prepare cost estimates covering the vision, identifying high-level requirements, and outlining the framework the aligns with the enterprise architecture guidelines. The second step is to determine the activity cost of each increment/release. The third step is to determine the activity-independent costs of each discipline needed for the entire project. The fourth step is to prepare the cost estimates for the project charter, incremental project scheduling, project plan, increment/release plans, deployment plan for each increment release and infrastructure plan for each increment/release. The fifth step is to add up the present value of all the increment/releases costs and derive a project budget. The sixth and final step is to calculate an initial increment release cost budget.

To estimate and budget for an increment/release the first thing to do is prepare cost estimates on the high-level requirements for the increment/release and for any changes in these requirements. The next thing is to prepare cost estimates on the framework and any changes to the framework for the increment/release. The third thing is to prepare cost estimates covering detail requirements, capital outlay (environment, hardware, networking, and technology), development cost for the build, resource layer costs (data source), testing costs, security plan, change and configuration management, and deployment cost. The fourth step is to prepare cost estimates for on-going costs (like training and travel) and administrative costs. The final step is to calculate the increment/release cost budget (based on increment/release end date and the increment/release estimate).

In estimating and budgeting for an iteration (of an increment) the first step is to identify the activities and the number of iterations in the increment/release. The number of iterations can vary increment/release to increment/release. The second step is to assess the cost of each iteration (an iteration's activity costs, on-going costs, and administrative costs). Then you assess the cost of growth iteration(s) to include any growth scenarios. The fourth step is to adjust the cost of each iteration so the total cost of all iterations plus the growth iteration(s) fall within the increment/release cost estimates. The final step is calculating the iteration cost budget.

The next routine task to assess and list the benefits the organization will accrue (asset-wise, function-wise, and financial-wise) by deploying the features in each increment/release.

Subsequently, the project/project manager will calculate the benefit elements (net cash flow, ROI, NPV, IRR, and payback period) of each increment/release. Then the project manager submits the paperwork for approval and allocation of funds.

So, to summarize, the project manager prepares the cost estimates, cost budget, and benefit elements for an increments/release (by functionality issues, functionality receipts, etc.) and for all increments.

It will help the project manager and corporate management decide whether to build, buy, reuse (internal), or use (external facility) the increment/release. It will also help the project manager/team accommodate any additional sub-functionality, additional infrastructure technology, and additional interface windows without additional costs by planning some iteration(s) costs with lower budgets.

So the primary goal concentrates on the financials of each increment/release. The main focus is on forecasting expenditure amounts and timing and income amounts and timing. So corporate finance can allocate the budget for an increment/release. The project manager/team are following primary Lean Principle criteria.

Since a primary criterion of the Lean Principle is to deliver value to a customer the project has to deliver all the agreed features of an increment/release. If the due date of the whole increment (Increment 1) isn't met, the increment/release would still be delivered on that day. The project manager would then plan a new increment (Increment E1) with the same team and assign it to add the leftover features. The project manager reviews the estimates and budgets for the remaining increments (by functionality) and prepares an estimate and budget for the Increment E1.

Since the other major criterion of Lean Principle is to increase of potential of the team, the project manager/team will try to increase the team's throughput (of increments) to balance increment costs against the constraints of the total project budget. So the project manager/team will constantly review and assess the current increment and iteratively lay out a new increment plan with a new timeline of the increments remaining (start dates and end dates), and project new estimates and budget figures covering the remaining increments.

And so that's how to follow the Lean Pattern in which we add value to the business by delivering working parts of the whole system and executing within allocated funds.

References:

- **Walker Royce**. Software Project Management: A Unified Framework.

- **Jason Charvat**. Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processors for Projects.

- **Craig Larman**. Agile and Iterative Development: A Manager's Guide.

- **Harold Kezner**. Project Management: A System Approach to Planning, Scheduling, and Controlling.

- **Barry Boehm** and **Ellis Horowitz**. Software Cost Estimation with COCOMO II.

- **James A. Brimson**. Activity: An Activity-based Costing Approach.

- **Mike Cohn**. Agile Estimating and Planning.

- **Paul Coombs**. IT Project Estimation: A Practical Guide to the Costing of Software.

- **O. Benediktsson** and **D. Dalcher**. Effort Estimation in Incremental Software Development.

- **Bruce Baggaley** and **Brian H. Maskell**. Practical Lean Accounting: A Proven System for Measuring and Managing the Lean Enterprise.
- **B. Boehm** and **D. Turner**. Balancing Agility and Discipline: A Guide for the Perplexed.
- **Mary Poppendieck** and **Tom Poppendieck**. Lean Software Development: An Agile Toolkit.
- **Michael L. George. Lean Six Sigma**: Combining Six Sigma Quality with Lean Production Speed.
- **Mary Poppendieck** and **Tom Poppendieck**. Implementing Lean Software Development: From Concept to Cash.
- **James W. Martin**. Lean Six Sigma for Supply Chain Management.
- **James P. Womack, Daniel T. Jones,** and **Daniel Roos.** The Machine That Changed the World: The Story of Lean Production.
- **James P. Womack** and **Daniel T. Jones**. Lean Thinking: Banish Waste and Create Wealth in Your Corporation.
- **Taiichi Ohno**. Toyota Production System: Beyond Large-Scale Production.
- **Rick Mugridge** and **Ward Cunnigham**. Fit for Developing Software.
- **Ronald Mascitell**i. Building a Project-driven Enterprise: How To Slash Waste and Boost Profits through Lean Project Management.
- **Peter Middleton** and **James Sutton**. Lean Software Strategies: Proven Techniques for Managers and Developers.
- **Jeffrey Liker**. The Toyota Way: Fourteen Management Principles from the World's Greatest Manufacturer.
- *IBM Redbooks*. SOA Foundation Service Creation Scenario.
- *IBM Redbooks*. Service Oriented Architecture and Web Services.
- *IBM Redbooks*. Patterns: Implementing a SOA Using an Enterprise Service Bus.
- **Eric A. Marcs** and **Michael Bell.** Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology
- **Eric Pulier, Hugh Taylor,** and **Paul Gaffney**. Understand Enterprises SOA.
- **Dirk Krafzig, Karl Banke,** and **Dirk Slama**. Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series).
- **Dan Woods** and **Thomas Matten**. Enterprise SOA: Designing IT for Business Innovation.

About the Author

Nandan Dasgupta is an independent system consultant.

dasgupta.nandan@gmail.com

Sanity, Strategy, Sustainability

– CONTINUED FROM PAGE 11

By including defect tracking, test-asset development and validation, VaS helps to deliver a common view of quality across disparate parts of an organization during the entire test and integration lifecycle. VaS provides management-level reports on key testing and integration metrics and helps to incorporate rich diagnostic capabilities such as software analysis, testing automation, and management. Verification-as-services raises the visibility of testing and integration so management and individual technical staff members can see concrete results from the increased focus on testing.

Many embedded software development teams have a pressing need to integrate and verify code from multiple sources, yet lack the time and resources to do it. Verification-as-services lets these teams focus on development, while simultaneously enabling a rigorous process for testing and measurement.

Implementing a sustainable verification platform can seem daunting. Fears about potentially disrupting the development work in progress keep many organizations stuck in the past. Still, relying on outdated test and verification methodologies is straining them. It's perpetuating a dysfunctional development process characterized by product quality problems, unpredictable delivery dates, and a climate of ongoing operational stress.

Verification-as-services, on the other hand, offers a new approach that addresses the new challenges of embedded development. It's a scaleable approach that lets teams start small and focus on specific components first then expand test coverage as needed. VaS offers engineering organizations the opportunity to eliminate most manual testing and automate integration, efficiently support multiple product releases, optimize software reuse, and effectively manage their software suppliers. It's a strategic solution that frees organizations from the day-to-day management of software test and verification. The benefits of VaS extend well past engineering. The decision to embrace this new approach will be a strategic business move to improve market competitiveness, enhance organizational effectiveness, and achieve the on-time delivery of higher-quality software products.

About the Author

Mark Underseth founded S2 Technologies in 2000, bringing over 20 years of experience in the design of embedded communication devices. Prior to founding S2 Technologies, he was the VP of engineering and responsible for all software development at Mitsubishi's Mobile Communications Technology Center in San Diego. His primary focus was on the development of the world's first single chip dual-mode IS-136 handset. Mark received a BA in computer science with an emphasis in mathematics from Point Loma College, and an MS in computer science from San Diego State University. He has been awarded four patents and has three patents pending in the area of embedded systems.

AJAXWORLD™

CONFERENCE & EXPO

Decision Year for RIAs

March 18-20, 2008 New York City

Rich Internet Applications:

AJAX • JAVAFX • SILVERLIGHT • ADODE AIR • FLEX • MASHUPS

Developers and managers are by now familiar with the usability benefits of user-centric Rich Internet Applications. The industry has matured, giving adopters the confidence to proceed. But many are still unsure whether AJAX is the answer, or whether one of the leading RIA alternatives might be a more suitable platform.

2008 will for many enterprises be the Decision Year for their choice of RIA platform, and AJAXWorld Conference & Expo 2008 (March 18-20, 2008) is the place to be if you want easy access under one roof to a welter of resources, ideas and examples to help you decide.

Because it's easy to misjudge priorities and invest in the wrong technology, AJAXWorld will help you invest in the right one for your particular business. It will help you find the optimal balance between business needs, technology potential, and enhanced user experience. AJAXWorld is where you can find what you're going to need for business survival and Web success in 2008.

Who Should Attend?

- CTOs & VPs of Engineering
- Directors of Technology
- Sr. User Interface Architects
- Front-End Engineers
- VCs & Industry Analysts
- Directors of Business Development
- Software Engineers
- Senior Architects
- Application Programmers & Software Developers
- Project Managers
- Web Programmers & Designers
- Companies & Organizations that need to stay in front of the latest Web technology

REGISTER TODAY AND SAVE!

www.ajaxworld.com

Using Integration Appliances To Reduce Complexity in SOA and Increase Business Value

BY CHANDAR PATTABHIRAM

Integration professionals have realized that point-to-point integration dramatically increases the complexity of building and maintaining corporate business processes. One manifestation of this is the increasing popularity of Service Oriented Architectures. To achieve maximum benefit from SOAs, it's important to carefully consider architecture and design principles before picking technology and working on practical examples.

In a properly factored design, integration appliances can greatly reduce cost while simplifying implementation and ongoing maintenance. Appliances can also help insulate companies from the inevitable changes common to any integration system by minimizing supporting infrastructure.

Barriers to Integration Success & the Original SOA

In the mid-90s, commercial integration packages were used to build large cross-system business processes for the first time. However, the limitations of first-generation technology became a serious barrier to success:

1. **Lack of scalability**. Integrating every system into every other system results in complexity that grows as the square of the number of endpoints. This clearly won't scale. The introduction of hub-and-bus models reduced these connections, with all of the data flowing through a central hub and repository.

2. **Reliance on low-level APIs and adapters.** When all of the data for a given system flowed through a central point, designers could think about reducing the amount of work required to maintain the endpoints. Even moderately complex systems have dozens of endpoints and the bulk of IT administration effort involves maintaining connectivity as the endpoints change. For example, SAP comes out with a new version of software, Oracle buys PeopleSoft, the mainframe payments system moves to a Unix server or vice versa.

   Early integration systems were usually tied directly to low-level APIs at the endpoints using adapters. Unfortunately, standards were few. In 1995, even ODBC didn't always work consistently. This resulted in large numbers of adapters having to be co-located with the target systems — at least one per application type and often more — to deal with multiple versions of an ERP system. Every time the endpoint changed, that interface had to be reworked. Worse, all of the business logic behind it was often tightly tied to the adapter, and it also had to be rewritten or ported. This led to the idea of a higher-level abstraction than a programming API.

3. **Proliferation of abstract services**. The concept of using abstract services, rather than low-level APIs, was an improvement, but it, too, has a serious weakness. The number of interfaces that are services isn't really any smaller than the number of physical connections to end systems. By creating a service, you get a better abstraction, but there are still too many of them. Worse, it's often the case that a well-designed service touches more than one underlying physical application. This creates the potential for overlapping services, data duplication, and data inconsistency. Again, the design doesn't scale.

In response, IT professionals embraced SOA. The central concepts of the original SOA were simplified connectivity, service abstraction, and data mastering. Unless all of these ideas are present in your SOA design, you won't derive maximum benefit from the resulting system. Conversely, a true SOA not only insulates you from underlying change, but lets you reuse the services.

The Key to SOA Implementation: Separation for Success

An SOA can be implemented in many different ways, using a variety of tools and technologies, including Web Services, middleware, frameworks, and integration platforms. Insight into an overriding SOA trend, however, may help to minimize your implementation costs and efforts. Integration tools were originally undifferentiated; there was one tool for every connection to every system. As the SOA concept evolved, the architecture differentiated into high-level categories with specific structures and functions (see Figure 1).

1. **Business process**: This SOA layer makes use of services and embodies those business processes that span multiple services such as meta-business processes, the basis of BPM and workflow.

2. **Directory services:** Common services such as entitlements and service directories plug into the service layer to insulate the overlying business process from changes.

3. **Existing applications:** The underlying programs that actually do the work.

4. **Data synchronization:** These integration components are used for building and maintaining parent/child data relationships.

These components have different functions and very different structures. They are often best constructed using different technologies.

Selecting the Right SOA Technologies

The complexity of an SOA project can be greatly reduced by choosing appropriate tools. You probably wouldn't build your purchase order system on top of LDAP. You also shouldn't use a BPM tool for data integration.

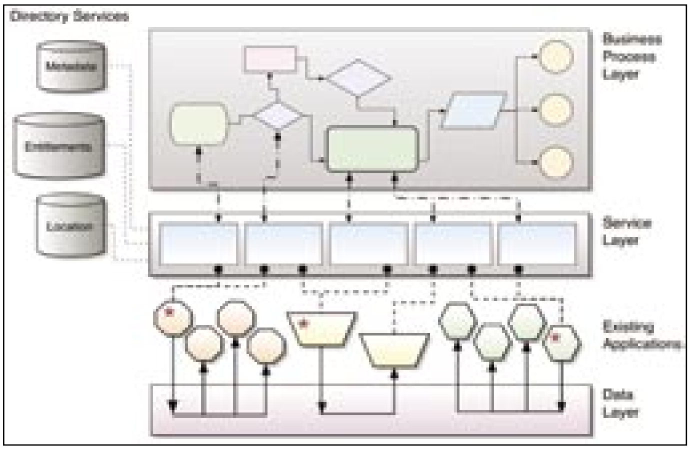


Figure 1: An efficient SOA

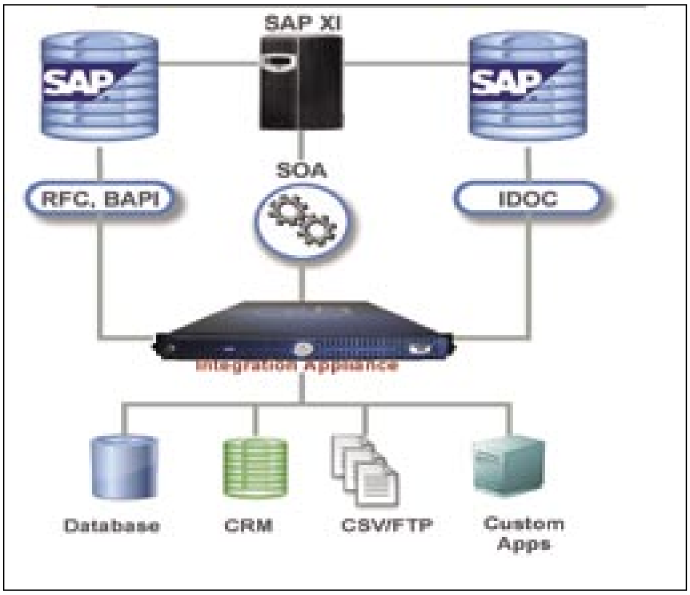


Figure 2: A major global CPG maker is enabling SOA and accelerating SAP adoption using integration appliances.

Good BPM tools are very expressive and abstract. They think in terms of end-to-end business collaborations that often involve multiple individual transactions done by humans. BPM tools are ideal for building high-end workflows containing complex business semantics and implementing best practices across multiple groups of users. They are also complex to learn, difficult to implement, and costly to maintain. If you really need what they have to offer, however, they're worth it.

In situations without complex manual workflows, an integration appliance is often more cost-effective, quicker to implement, and easier to maintain.

For example, a major global CPG maker is enabling SOA and accelerating SAP adoption using integration appliances. The company's first implementation of appliances was to replace a legacy middleware infrastructure in the synchronization of purchasing, foreign exchange, and retail data between ERP and other systems in approximately 20 countries. The results of this approach are quite impressive: integration appliances yielded an ROI payback in fewer months, reduced total cost of ownership by 74%, and delivered projects in 30 days or less. As the manufacturer migrated to an SAP-centric application footprint across its 170 subsidiaries in 50 different countries, it adopted a SOA-based enterprise integration framework – SAP instances are integrated with each other using SAP's own middleware platform, integration appliances are used to extend SAP to other applications worldwide, and a SOA framework has been implemented to bridge the SAP middleware with integration appliances (see Figure 2).

Another leading online marketing services provider is using integration appliances to implement a SOA solution for business-to-business communication. The company has adopted salesforce.com as its CRM solution and uses Microsoft Dynamics for back-office ERP functions. It uses integration appliances to expose order processing services to its vendors; these services are standards-based abstractions of discrete ERP and CRM operations like order creation and order visibility. Vendors communicate with the integration appliances to access these services in a real-time manner; the appliance in turn handles the native connectivity to the back-end applications as well as the data mapping, workflow, and error-processing logic (see Figure 3).

Using Integration Appliances in SOA

Even the best planned SOA project has its complexities and requires a broad skill set to implement and maintain. The thinking that originally led to SOA was focused on reducing that complexity. Complexity reduction pays dividends each time a change is made anywhere in a SOA, as the number of moving parts affected by the change is directly related to cost, time, and difficulty of implementation.

Integration appliances are useful because they're simple. They require minimal support infrastructure and are highly repeatable and secure. An appliance that can be accessed only via the network can, by definition, be configured, operated, and maintained from anywhere with a secure network connection. Appliances are simple because no infrastructure preparation is required beyond electricity and an IP connection.

Software solutions require an infrastructure composed of host hardware, network hardware, an OS, JVM, patches, RDMS, security software, and management software. Since software makers are constantly "improving" their products, the odds that any set of infrastructure servers are actually identical, however carefully standardized, is almost zero. Appliances, on the other hand, can

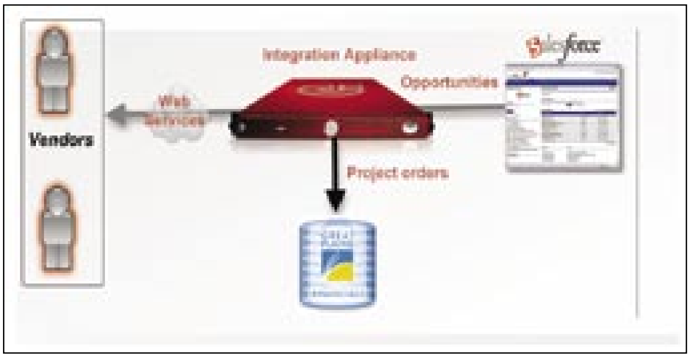


Figure 3: A leading online marketing services provider is using integration appliances to implement a SOA solution for business-to-business communication.

easily guarantee that form, fit, and function are identical because all the software and firmware in the box is under a single release control program. They are, in effect, one large software, hardware, and firmware distribution.

Because appliances aren't general-purpose, their components can be optimized for the task they perform. Similarly, there's no need to provide user-level access to internal components, such as RDMS, RAID arrays, OS functions, etc. They come preconfigured and can be managed by the appliance itself. All the user needs to do is configure the device for exactly the task it must perform.

As network devices, appliances can be placed wherever they are needed, and easily managed and upgraded. Local IT presence isn't required. Indeed, many appliance users manage worldwide implementations from a single center of excellence.

Integration appliances fit into three places in the SOA: they are the most logical solution for data integration, they are often the best way to integrate various directory services, and they can be used to construct the services exposed to the workflow/BPM layer:

1. **Data integration**. By avoiding the complexity of manual process integration, appliances can also avoid the bane of every integration project: adapters. Adapters allow rich interactions with endpoints and users at the BPM level. They're also hard to maintain, hard to upgrade, and expensive. With recent improvements in standards, adapters are usually unnecessary. In simple cases, direct data integration suffices. In more complex cases, the service can often be built directly on the target applications.

   We have already established that solutions using adapters don't scale. There may still be a few cases in which the use of adapters is unavoidable, but not in data integration. By combining the advantages of an appliance with the elimination of adapters, the master data integration scheme described above becomes simple and cost-effective enough for real projects.

2. **Directory services**. Directories for entitlements, service discovery, etc. need to present their information in a single format through a unified interface. Though simple in theory, this is almost never the way real corporate directories look and act. This is really a data integration problem in disguise. As such, an appliance can be a very good solution for normalizing directory access.

3. **Service presentation**. The complexity of a service is intermediate between that of a data interface and an application. A properly designed data integration appliance isn't suitable for manual workflow but is capable of holding and representing enough state (via BPEL, for instance) to serve as an intermediary between applications and the workflow/BPM layer. Since the most common place from which to access a directory is the service layer, it's possible to build a uniform and easily managed infrastructure that abstracts all but the services themselves away from the developer of the business process.

Conclusion

The success of an SOA depends on how well complexity has been reduced. Construction of large, robust integration projects that have a long, maintainable lifetime requires the architect to minimize complexity at every step. Rich agile business processes are where the value of the completed integration project lies. By forcing complexity out of the service layer, integration appliances allow more resources to be applied to the business layer where the ultimate payback for the company can be found.

Resources

- An Integrated Service-oriented Development Platform for Realization of e-Business Systems: http://whitepapers.techrepublic.com.com/whitepaper.aspx?docid=286071
- The Integration Journey - A field guide for business integration with SOA: http://www.cio.co.uk/whitepapers/index.cfm?whitepaperid=5348
- David Linthicum. Real World SOA: http://weblog.infoworld.com/realworldsoa/. ■

About the Author

Chandar Pattabhiram is vice-president of product marketing at Cast Iron Systems, a fast-growing provider of integration appliances. Chandar is responsible for strategy, messaging, pricing, and field and channel enablement. Previously, in a senior marketing role at Jamcracker, he managed the integration and provisioning product lines. Chandar also was a manager of the electronics & high-tech group at Andersen Consulting (Accenture). He has a master's degree in business management from the University of Texas and a bachelor's degree in mechanical engineering from PSG College of Technology.

chandar@castiron.com

The SaaS Superhighway

The global convergence of the telecom & software industries with telecom on top

WRITTEN BY ASH MASSOUDI

For years, the telecom industry has been aggressively expanding into IT services to offset declining revenue from traditional voice and data services.

Today telecom's infrastructure-focused IT services such as computer hosting are already commoditized. Telecom companies are trying to figure out how to climb up the IT stack towards application-focused software services where the business value is more direct and the margins higher. As it turns out, the Application Service Provider (ASP) model proved that turning licensed, single-tenant, traditional applications (such as those from Oracle) into a managed service doesn't have the required economies of scale. More importantly, the ASP delivery model is a stretch for the subscription/usage-based telecom providers.

On the other hand, the relatively nascent Software-as-a-Service (SaaS) model provides both the economies of scale and delivery model that's a natural match for the telecom industry. While the SaaS model is an obvious underpinning for telecom and software convergence, what is not so obvious is the makeup of the enabling technologies and the exact role a telecom provider has to assume to promote a value chain that can scale to tens of billions in new annual revenue.

The first-mover telecom provider that correctly identifies a) the key enabling technologies, b) the required value chain and c) its role in the value chain will be among the top beneficiaries of the convergence. Before discussing the enabling technologies and the make up of the value chain, let's see what it takes for the customers of business applications to endorse the pay-as-you-go convergence with their wallets.

Customers Need the Power of IT in the Hands of Business

There are four major problems with traditional business applications that have made it impossible to align IT and business: a) the high total cost of ownership (TCO) diverts well over 60% of IT budgets to application maintenance and upgrades instead of business innovation, b) the difficulty of integrating across application silos hinders business automation and efficiency, c) the lack of flexibility in the often over-provisioned underlying hardware and software infrastructure increases both exposure to risk and project costs, and d) the lack of agility in responding to new business requirements disconnects IT from the real-time realities of business.

The enterprise customer simply needs the power of responsive IT in the hands of its business leaders.

The SaaS model combined with Service Oriented Architecture (SOA) and platform innovations on top of SOA has the potential to address all the shortcomings of the traditional software model. However, so far the SaaS model has proven effective mostly for small and medium-size enterprises (SMBs) and non-core applications such as customer relation management (CRM) and human resources (HR).

There are two barriers that have prevented mainstream adoption of SaaS by large enterprises:

1. **Limited ability to customize**. The SaaS platform used by existing vendors brings multi-tenancy. While this innovation maximizes the operational efficiency of the SaaS vendor, it currently limits the customer's ability to customize the application business logic; large enterprises require customization to address changing business requirements and gain competitive advantage especially in the area of core business functions.
2. **Difficulty in integrating SaaS solutions with ERP systems inside the firewall**. The SaaS platforms used by existing vendors, although Web Service-enabled, provide no innovative inside-out integration capability (i.e., from SaaS platform to other ERP applications inside the firewall).

As you'll see next, the technological convergence of SaaS, SOA, and virtualization can overcome the barriers of SaaS adoption for large enterprises and enable telecom providers to target the SaaS market starting from the Global 1000 end of the market spectrum.

The Enabling Technologies

Network services such as Internet data centers, global network access, hardware, and OS infrastructure are offered as com-

Rich Web Technologies for Enterprise Web 2.0 & Social Applications

6TH INTERNATIONAL

AJAXWORLD™ CONFERENCE & EXPO

March 18-19-20, 2008
New York City
www.ajaxworld.com

Join More than 10,000 Early Adopters Who've Already Attended!

Come & Learn How to Achieve the Optimum Balance Between Business Needs, Technology Potential, & Enhanced User Experience!

Let AJAXWorld Provide Your Business Blueprint for Future Web Success. Doing business on the new Web is 99% about innovation and optimizing the user experience, and AJAXWorld Conference & Expo is 100% about the tools that encourage and support a superior user experience - the tools for business survival and Web success in 2008 and beyond.

Reasons to be there...

- You will leave AJAXWorld Conference & Expo 2008 East with resources, ideas and examples that you can apply immediately.
- Whether you're going to be using the technology to enhance an existing Web application or to migrate a client/server application to the Web, AJAXWorld will help you get up to speed more quickly.
- Attendance will improve your company's ability to adapt and prosper in the Web 2.0, user-centric environment.
- Whether you need to work with a designer or do the work yourself, front-end engineering makes more sense after you return from AJAXWorld!

The Real-World Questions That Will Be Answered by This Yearts Faculty!

- AJAX development IDEs: what are the alternatives?
- When it comes to accessibility, how do I know what's required?
- How can I make AJAX applications that easily go offline?
- How do you make a secure AJAX application?
- Are mashups really contributing to the experience of the Web?
- What's the difference between a mashup and a composite application?
- Am I better off using an AJAX framework, a toolkit or just coding?
- Is AJAX commoditized or will it be soon?
- Will AJAX be standardized in the form of widget APIs or declarative markup?
- What AJAX development tools are available for visual development today?
- How should you handle Web analytics and metrics for Rich Web sites?
- Do we really need JavaScript 2.0?
- How do you scale an AJAX-based design to a very large application?
- How can I harness the energy of my customer base online?
- On what timeline will AJAX skills become commoditized?

Technical sessions will cover challengers to AJAX such as new innovations like Silverlight and JavaFX. We will be dealing with performance and browser compatability issues, showcasing viable strategies for performing unit/stress testing on an AJAX application, and how to apply user interface patterns and user experience designs to your AJAX project.

Every aspect of AJAX and rich-Web alternatives will be covered, and the emphasis throughout will be on actionable insights, real-world case studies, and up-to-date technical information and advice.

Who Should Attend...

- CTOs & VPs of Engineering
- Directors of Technology
- Sr. User Interface Architects
- Front-End Engineers
- VCs & Industry Analysts
- Directors of Business Development
- Software Engineers
- Senior Architects
- Application Programmers & Software Developers
- Project Managers
- Web Programmers & Designers
- Companies & Organizations that need to stay in front of the latest Web technology

2008 Call For Papers Now Open!

AJAXWorld offers the undisputed best platform to position your company as a leading vendor in the fast-emerging marketplace for AJAX and Enterprise Web 2.0.

REGISTER TODAY & SAVE

Must-See Speakers

BOLOKER	WEI	FERRAIOLO	CROCKFORD	PELED	PELLER
IBM	Nexaweb	OpenAjax	Yahoo!	Gizmox	IBM

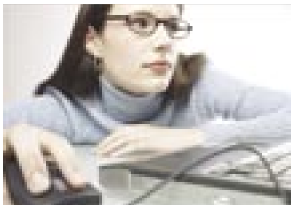
SCHALK	GUPTA	MIRAGLIA	STAGNER	POLLACK	REGAN
Google	Sun	Yahoo!	Microsoft	Ruby Users	Sonoa

AJAXWorld 2008 East - Tracks

- **Track 01:** Enterprise AJAX
- **Track 02:** Rich-Web Case Studies
- **Track 03:** Enterprise Web 2.0 & Social Applications
- **Track 04:** Enterprise Web 2.0 Mashups
- **Track 06:** iPhone Developer Summit
- **Track 06:** Diamond Track

Introducing in 2008

AJAXWORLD™ SECURITY BOOTCAMP



ajaxsecuritybootcamp.sys-con.com

March 18, 2008 New York
October 20, 2008 CA

iPhone Developer Summit

iphonedevsummit.com

March 19-20, 2008 New York
October 20-22, 2008 CA

Call Now for Sponsor & Exhibitor Opportunities
Space Is Limited 201-802-3021 or carmen@sys-con.com

CALL 201-802-3020 OR E-MAIL EVENTS@SYS-CON.COM

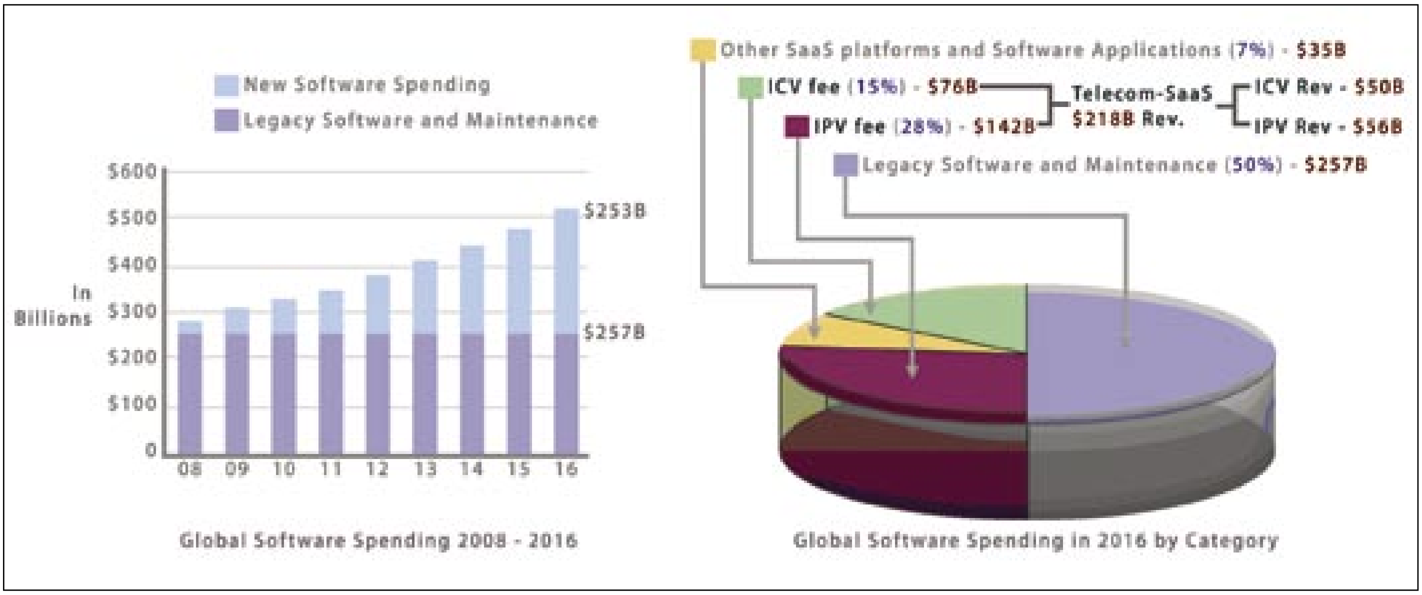


Figure 1:

modities by telecom providers and some specialized and smaller network service providers. On the other hand, managed services such as security, QoS, and billing expertise are among the valuable assets that telecom providers can bring to the SaaS table from a technology perspective.

To accommodate the creation, customization, integration, and distribution of SaaS applications, some key technologies are required. These technologies go beyond the core competency of telecom providers. These technologies need to provide functions similar to middleware functions. However, they need to be much less complex than middleware to accommodate customer requirements for reduced TCO, ease of integration, and increased agility. Besides, middleware technologies aren't built for multi-tenancy and so do not lend themselves to the SaaS model.

The software platform technologies needed to enable agile SaaS solutions for SMBs through large enterprises are addressed under the four categories below:

1. ***GUI Layer:*** This is also a commoditized layer. A good example here is Adobe Flex. It is a rich Internet client with the ability to consume Web Services out of the box and does most of its processing on the client machine so it provides a good choice for SaaS solutions.
2. ***Model-driven Service Composition as a Service.*** A new model-driven componentization technique called SOP for Service Oriented Programming (http://en.wikipedia.org/wiki/Service_Oriented_Programming) has made it possible to converge SaaS, SOA, and virtualization platforms into a single model-driven platform and so drop the SaaS penetration barriers of large enterprises (http://soa.sys-con.com/read/467329.htm). This model-driven service composition platform is multi-tenant and provides all the functions of SOA middleware without the associated complexity. This is the single most important enabler for the telecom-SaaS technology stack.
3. ***Service Management & Governance.*** In the model-driven telecom-SaaS infrastructure, every exposed and reusable component of a SaaS application is a software service. This means that service management and governance functions must be native to the Service Composition platform.
4. ***Service Exchange Infrastructure***. Think of this as being something between the eBay and the Yellow Pages of services where SaaS application component vendors can catalog their offerings as a set of software services and be reviewed and rated by customers and the telecom provider.

The Winning Value Chain

To generate revenue worthy of attention in a trillion-dollar industry, telecom providers will have to replace the "SaaS country

"While the SaaS model obviously underpins telecom and software convergence, what is not so obvious is the makeup of the enabling technologies and the exact role a telecom provider has to assume to promote a value chain that can scale to tens of billions in new annual revenue in less than a decade "

road" approach promoted by the single SaaS vendor with a value chain that results in a "SaaS superhighway" capable of generating hundreds of billions in revenue. The successful telecom provider views its major role in the SaaS ecosystem as a conduit to enterprise customers and system integrators (SIs). The goal is to create a value chain that turns the telecom provider into the most scalable channel for SaaS offerings in the shortest time possible. This requires a series of key partnerships as opposed to building everything, or acquiring every one-off SaaS application vendor and ending up with a hodge-podge of underlying platforms that are hard to integrate and consolidate.

On the consumer side of the value chain, most SMB and Fortune 2000 customers prefer to acquire end-to-end solutions through a single vendor whenever possible. From this perspective, telecom providers with their vast enterprise customer base are in a unique position to act as distribution channels for SaaS solutions. Also, up to now, SIs had to rely on vendor-specific application platforms to deliver solutions. SIs have been looking for a vendor-neutral platform to capture their domain knowledge and increase reusability across clients. Because of this, SIs are likely to become a conduit for the telecom-SaaS offerings.

On the platform producer side of the value chain, the successful telecom provider forges partnerships with the most innovative and independent software platform as a service vendor (IPV) to provide real-time service composition and assembly, service management, and governance and a service exchange infrastructure. The partnership should require the platform partners to integrate to the telecom provider's managed services such as security, billing, storage, service assurance, and QoS services. Since the platform-as-service offerings are hosted, the telecom provider can further enhance its business by selling data center and computer hosting services to these partners. Although this sounds like double dipping (and it is), it provides tremendous scalability and performance benefits for everyone due to the proximity of all the platform services. On the application component producer side of the value chain, traditional ISVs and independent new entrants throughout the global economy can use the service exchange infrastructure to catalog their application-specific components as services. The telecom provider can also sell data center and computer hosting service to these independent application component vendors (ICVs).

Finally, telecom-SaaS providers bring the software platform-as-a-service with a catalog of application services to end customers and SIs. The customer chooses some of the application components out-of-the-box and uses model-driven platform services to rapidly assemble highly customized and integrated solutions on-demand using internal IT staff or SIs.

Unlike traditional enterprise software license sales, the telecom provider's sales force won't require any specific domain knowledge for selling telecom-SaaS offerings. Instead, it's focused on exposing the customer to the telecom-SaaS and its new economics through select use-cases. Then, the customer can easily evaluate any of the offerings and use what it needs on a pay-as-you-go basis instead of the elaborate enterprise software acquisition process common today. Each ICV is responsible for providing standardized marketing collateral in the form of recorded presentations and demos to provide domain-specific information. The telecom-SaaS platform enables a pull model where the customer pulls what it needs when it needs it as opposed to the push sales model common today. The pull model is enabled through the service exchange and can utilize social networks to connect customers across verticals and common interests.

The Size of the Opportunity & Revenue Distribution

Extending an estimated CAGR of 7.7% growth in software spending through 2016, as IDC assumes at least until 2010, new software spending will exceed $250 billion in 2016 while total software spending reaches $510 billion. Note that the IDC growth estimate in 2005 didn't consider the additional growth factor of telecom and software convergence through the telecom-SaaS offering. Considering the exponential decay of traditional software's share in new software spending and the new economics of solutions, $218 billion in software spending will flow through the telecom industry as revenue. Out of the $218 billion in revenue, approximately $50 billion of the $76 billion ICV fees paid by the customer will be passed to ICVs assuming the telecom provider will earn a 35% channel fee for sales and distribution. Also, approximately $56 billion of the $142 billion in IPV fees paid by the customer will be passed to IPVs assuming a 60% channel fee for sales, marketing, and distribution.

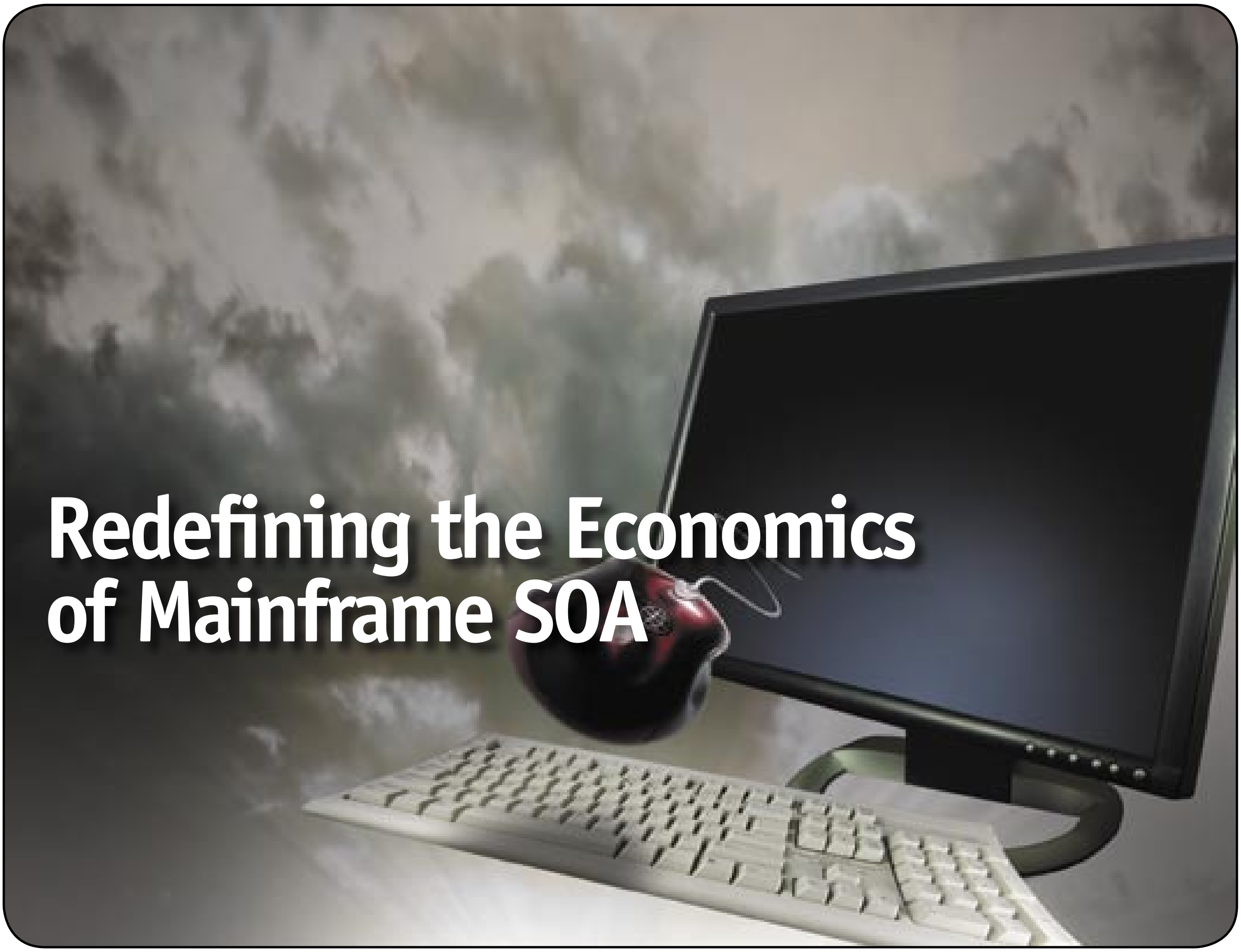
The convergence of the telecom and software industries through the telecom-SaaS distribution channel will result in a global wave of IT growth. This wave will create major global opportunities for independent software vendors in providing platform, application component, and packaged composite applications as services. In addition, it will fuel the market for SIs by bolstering demand for professional services for on-demand application assembly and reassembly.

SMB and Fortune 2000 enterprise customers are the biggest winners in convergence. The global nature and the large scale of the telecom-SaaS results in the rapid commoditization of application service components. The on-demand composition and assembly of application services also increases the responsiveness of businesses to changing business requirements.

By now, you maybe wondering, who makes the best early adopter for the telecom-SaaS offerings? I'd have to say the telecom provider who wants to be at the forefront of convergence. ■

About the Author

Ash Massoudi bootstrapped NextAxiom Technology, Inc. (NxA) in 2000 with only $500,000 of external capital as seed money. As CEO of NextAxiom, he led the conception and delivery of an innovative model-driven service-composition platform to a customer base that includes some of the largest energy and utility companies in the world. Prior to founding NxA, Ash was the director of integration technologies at PeopleSoft. In that role he led the development and delivery of a comprehensive XML-based message-oriented infrastructure as part of PeopleSoft 8.0 to hundreds of Fortune 2000 companies. He has over 15 years of experience in business software spanning client services, engineering, product management, business development, sales, and alliances.



Redefining the Economics of Mainframe SOA

BY GREGG WILLHOIT

Next-generation middleware exploits IBM System z specialty engines, redefining mainframe total cost of ownership and spurring expanded legacy participation in Service Oriented Architectures

Of all the wonders Service Oriented Architecture has wrought in the business world, one of the most valuable has been to unlock legacy data, applications, and processing resources for new and profitable use. Residing on mainframe platforms, such assets have been notoriously tricky and costly to change and adapt to new, typically web-centric, purposes. On the other hand, they often excel at the critical functions they have provided, typically for decades, and are highly reliable. Replacing them outright would be extremely costly as well as disruptive to operations that are commonly mission-critical to an organization.

The rapid adoption of SOA in business and government organizations has radically changed this picture, and has opened the door to almost unlimited ways of mining the riches of the legacy assets contained in those organizations' mainframes. It has dramatically altered the perception and role of the mainframe, making it considerably more capable of participating in loosely coupled, standards-based cross-platform environments. To be more agile and competitive, organizations with tremendous investments in legacy mainframes are turning to SOA to provide more infrastructure flexibility with reduced development cycles and lower costs. With a heavy reliance on the industry-standard Simple Object Access Protocol (SOAP) and Web Services Description Language (WDSL), SOA's role in enabling the mainframe's participation in modern distributed and web-centric applications is largely a function of the middleware software. However, IBM has been no passive observer in keeping its mainframes a viable component in enterprise IT and the emerging services-based architectures. Big Blue has made considerable reinvestment in its mainframe operating system software and has introduced several innovations in the form of "specialty engines" to its latest System z architecture. These specialty engines, in combination with advanced middleware, can result in a powerful symbiosis capable of delivering dramatic increases in mainframe SOA and data connectivity performance, as well as considerable reductions in mainframe MIPPs utilization and software costs. The cost reductions and the potential for these specialty engines to enlist the mainframe in SOA and business process optimization initiatives amount to more than just incremental advancements. For businesses that have been forced by growing mainframe software costs to consider the risks of switching to distributed platforms, it is nothing short of revolutionary.

IBM characterizes its new System z specialty engines as processors that can help users expand the use of the mainframe for new workloads, while helping to lower Total Cost of Ownership (TCO). These specialty engines are:

- **System z9 Integrated Information Processor (zIIP**) — Processing engines that are capable of executing work in a restricted environment. The zIIP's execution environment accepts eligible work from specialized z/OS code, which manages and directs the work between general-purpose processors and zIIP engines.
- **System z Application Assist Processor (zAAP)** — Specialized processing engines that provide a strategic z/OS Java execution environment, as well as — in the latest development — to provide some XML parsing.
- **Integrated Facility for Linux (IFL)** — Processors dedicated to Linux workloads.
- **Internal Coupling Facility (ICF)** — Allows multiple z/OS Logical Partitions (LPARs) to share, cache, update, and balance data access in a parallel sysplex.

This article focuses on the zIIP and zAAP specialty engines insofar as they can be leveraged by mainframe middleware used in developing SOAs and business process management applications. The salient issue is that mainframe middleware utilizing lower-cost zIIP and zAAP specialty engines will lower TCO for SOA workloads while simultaneously freeing up capacity on the significantly more expensive general-purpose processors. Unless well informed about the workings of IBM's System z and its latest advances, enterprise architects developing SOAs may overlook integration middleware that exploits these specialty engines, limiting the full potential for mainframe involvement and saddling their organizations with less than optimal performance and value.

The Thinking behind IBM's New Specialty Engines

Mainframe systems have a reputation for being expensive. Historically the common tendency was to attribute this to hardware pricing; however, mainframe shops know that today it's mainframe software costs that have the potential to break the bank. The cost challenge for most such organizations lies in managing the cascade of incremental software charges from their independent software vendors tied to MSU increases (an MSU is an IBM term used to measure the amount of processing work a computer can do in one hour). An organization might project doubling its mainframe MSUs in five years only to subsequently discover that the actual rate of business growth for that timeframe falls short of those expectations. If the organization had pre-purchased the MSUs to accommodate the projected growth, they'd have immediately incurred software maintenance charges from all the ISVs — charges for capacity they turn out not to have used.

IBM first attempted to address this aspect of mainframe TCO by designing a means to govern MSU capacity electronically using microcode to "throttle" the actual full processing capacity of a mainframe computer. This approach let IBM sell a customer a CPU that could run at, say, 5,000 MSUs and set it to run at only 224 MSUs. That customer could purchase upgrades later at incremental rates on an as-needed basis as business growth required or budgetary constraints allowed. This way IBM got additional revenue and its customers benefited from non-disruptive upgrades that let them keep a tight rein on software charges.

With the new specialty engines, which IBM has made available as architecture facilities on the System z platform, IBM has provided another way to improve mainframe TCO. The zIIP and zAAP specialty processing units arose partly out of the realization that certain types of workloads could be more effectively handled outside the general-purpose processor (GPP). Running Java on a mainframe is highly processor-intensive. Having a specialty engine (zAAP) dedicated to this kind of workload is strategic to IBM and its ability to reinforce WebSphere, its Java-based enterprise platform suite. By the same token, the zIIP specialty engine plays a role in IBM's DB2 strategy, making it easier and considerably more cost-effective for data-intensive applications such as Enterprise Resource Planning (ERP), Business Intelligence (BI), and Customer Relationship Management (CRM) to have parts of their queries directed to the zIIP.

Three interrelated factors make these new specialty engines special:

- The workload deployed on specialty engines is diverted from the GPP (depending on specialty engine availability), increasing the availability of the GPP to handle other workloads and reducing the need to buy additional capacity.
- Specialty engines are also exempt from the incremental microcode "throttling" imposed on the GPP. This means that any workload running on a zIIP/zAAP can run at full unrestricted speed for higher performance.
- The work dispatched on the zIIP/zAAP is not subject to measured usage charges, thereby reducing the overall measured MSU consumption and avoiding a cascade of associated ISV-related charges.

New DNA for Next-Generation Middleware

It's commonplace enough today to find mainframe integration middleware that lets mainframe data sources be virtually extended to support composite application development or provide SOAP wrappers for mainframe transactions to participate in a SOA. However, a new generation of mainframe middleware is just emerging that cunningly exploits the specialty engines in IBM's latest System z architecture to deliver significant performance enhancements for SOA and business process management while simultaneously reducing hardware and software usage costs. This capability requires sophisticated software development skills as well as intimate knowledge of the latest z/OS features.

To appreciate the craft underlying such commercially available middleware, let's explore the multitasking environment of the mainframe in more detail. Mainframe processors are dispatched as threads through Task Control Blocks (TCBs), but not all threads are created equal. Within z/OS, the Workload Manager (WLM) software prioritizes TCBs. WLM helps administrators control workloads according to such considerations as Quality of Service (QoS) and Service Level Agreement (SLA) goals. The z/OS also uses a lightweight, low-overhead kind of thread called an enclave Service Request Block (SRB). The zIIP specialty engine runs certain enclave SRBs that are zIIP-eligible. Those SRB workloads can run without the governing constraints put on the GPP and are free of the associated software charges. Very few middleware products sold today that support mainframe SOA were written to run in enclave SRB mode. They are TCB-based products incapable of exploiting the zIIP.

While most workloads running in an enclave SRB are zIIP-eligible, SVC (supervisor calls) can't be executed in an enclave SRB. This makes certain operating system functions such as I/O unavailable while in enclave SRB mode. What's needed for next-generation middleware is a hybrid SRB/TCB architecture that creates a pair of threads — that is, an enclave SRB and a TCB, with the ability to run most of the code in the enclave SRB to exploit the zIIP, with the execution of the SVC calls switched over to the TCB. This unique innovation enables maximizing the zIIP usage while avoiding any restrictions as to what the application can do. This architecture becomes the "DNA" building block of applications that expedite performance while minimizing cost.

A look at how this new class of software uses the zIIP engine may help illustrate the benefit of operating in enclave SRB mode. Consider the scenario of processing a mainframe Web Service. When a request comes into the mainframe for that Web Service, the first step will require translation of the Web Service names and operations into WLM service classes. Newer more evolved mainframe middleware can create an enclave SRB, which is eligible to be executed on the zIIP while also providing the ability to designate the percentage of the SOA workload that's off-loaded to the specialty engine.

Provided they understand the z/OS and the nuances of WLM operations, developers working on next-generation mainframe middleware can take steps to ensure that many workload components of SOA also are made zIIP-eligible, including much of the XML/SOAP messaging processing, Open Database Connectivity (ODBC)/Java Database Connectivity (JDBC), ADO.NET processing, TCP/IP processing, tracing, and security. These kinds of CPU-intensive workloads are where the mainframe platform shines, and they can now enjoy the full unrestricted processing power of the zIIP.

The Mainframe as a Driver of Business Process Optimization

In an ideal SOA implementation incorporating mainframes, leveraging assets is a bi-directional affair. That is, mainframe strengths — proven, high-volume transactional processing and data management — aren't merely subsumed into the advantageous characteristics of a standards-based distributed platform, but the distributed platform also partakes in the advantageous characteristics associated with the mainframe platform. Factor in exploitation of the new specialty engines, and the mainframe suddenly demands fresh consideration as the most logical as well as cost-effective platform for modern business process management and automation.

Consider the example of an organization that wants to use Web Services to drive a mainframe-based airline reservation Web portal with a high-transaction rate and a sub-second response time requirement. The overhead imposed by the XML-based SOAP parsing associated with the mainframe Web Services may actually prevent those transactions from occurring in less than a second. If successful, the construct will certainly be ungainly and the solution no doubt costly from an MSU perspective. However, if that XML processing can be offloaded from the governed GPP to a zIIP or zAAP, it can now avail itself of the unchecked processing power of the specialty engine. Moreover, the XML workload now no longer takes place on the measurable GPP, subject to software-per-MSU charges, thus dramatically increasing the cost efficiency of the overall mainframe solution.

Now let's say this airline reservation portal relied on Web Services built on mainframe screen logic (not uncommon in legacy mainframe environments), and that — besides XML parsing —Web Services orchestration is also needed. Java is the primary platform of choice for running industry-standard Business Process Execution Language (BPEL) for Web Services 2.0, which is the preferred way to orchestrate mainframe screen-based Web Services and provides for a top-down process-oriented approach to SOA. With next-generation middleware written to exploit these new specialty engines, organizations wanting to integrate mainframe assets into a SOA using BPEL orchestration (via a JVM) now have the ability to leverage the zAAP to run this critical component of the integration scheme.

No mainframe shop – large or small – can afford to ignore the potential cost impact of the new specialty engines. Those with smaller mainframes who are considering a switch to distributed systems to save on soaring software costs might well reconsider. For example, a medium-sized business that forecasts the need for 15 additional MIPs in an upcoming year to accommodate business growth might incur additional software fees of around $600,000 a month, or $7.2 million a year. If that company could move enough of its workload to zIIP and/or zAAP engines, it might find that it only needs to upgrade five MIPs and can reduce its projected software costs by nearly $5 million.

Changing Requirements for Architects

As is clear from earlier discussion of a hybrid SRB/TCB architecture as the new "DNA" for applications that leverage IBM's latest innovations, sophisticated development and deep knowledge of z/OS architecture is needed to design new-generation middleware that can leverage the specialty processors for SOA and BPEL implementations.

That's not to say, however, that if you are an IT architect and/or administrator planning such initiatives for your organization, you must have this expertise. You need only be informed enough to know what to look for as a requirement in commercially available middleware and integration software. Consider providers that have started adding new functionality to their products to take advantage of both the zIIP and zAAP specialty engines. Consider what percentage of SOA- and/or BPEL-related workloads will be able to be run on zIIP or zAAP engines in your applications using this middleware.

The landscape for SOA and BPEL has changed, and the new class of middleware available now can advance the value proposition of the mainframe platform, building on IBM's own innovation with an alternative approach that benefits mainframe customers and expands the future and role of the mainframe across the service-enabled enterprise. ■

About the Author

Gregg Willhoit is chief software architect for DataDirect Technologies (www.datadirect.com) and provides technical leadership for the Shadow mainframe SOA and data integration middleware products. He has architected and developed a wide variety of commercially successful software products including DBMS catalog management tools, MVS and DB2 performance monitors, and is the co-developer of the Shadow Direct ODBC-to-mainframe data product for DB2, IMS, VSAM, CICS, and Adabas.

gregg.willhoit@datadirect.com

Visit the *New* www.SYS-CON.com Website Today!

SYS-CON MEDIA
intel
SYS-CON Intel Developer FORUM
Today's Top IT Solutions Providers!
.NET 2005

The World's Leading *i*-Technology News and Information Source

24/7

FREE NEWSLETTERS
Stay ahead of the i-Technology curve with E-mail updates on what's happening in your industry

SYS-CON.TV
Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!
Read web logs from the movers and shakers or create your own blog to be read by millions

WEBCAST
Streaming video on today's i-Technology news, events, and webinars

EDUCATION
The world's leading online i-Technology university

RESEARCH
i-Technology data "and" analysis for business decision-makers

MAGAZINES
View the current issue and past archives of your favorite i-Technology journal

INTERNATIONAL SITES
Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

IT Solutions Guide
Information Storage+Security Journal
JDJ
Web Services Journal
.NET Developer's Journal
LinuxWorld Magazine
Linux Business News
Eclipse Developer's Journal
MX Developer's Journal
ColdFusion Developer's Journal
XML Journal
Wireless Business & Technology
Symbian Developer's Journal
WebSphere Journal
WLDJ
PowerBuilder Developer's Journal

SYS-CON MEDIA *The World's Leading i-Technology Publisher*

When You Need to Cancel a SOA Project

Knowing when to pull the plug

BY DAVID S. LINTHICUM

Many SOA projects are created out of hype, not need. Clearly many enterprises are "managing by magazine" and are more concerned about doing something cool rather than doing something helpful. You know the difference, and I'm sure there are both types of projects in your organization today.

Indeed SOA has become popular, but not in a good way. Tactical, on-off projects are sprouting up all over the place with poorly defined values and strategic direction. Thus, they bring very little to the architectural party and could be making the enterprise take a few steps back.

Bad SOA, even though it's SOA, is still bad. These projects can typically be found around what I would call VDA or Vendor Driven Architecture-type projects. You know them by the name of "the ESB project" or the "SOA governance project," and thus they are really technology-driven versus business- or value-driven. These projects are bad for the enterprise and bad for SOA.

Should you cancel a SOA project? Here are some things to consider.

First, no strategic vision around how that projects fits in the overall enterprise architecture. This is the Big One. If there is no roadmap, no plan, no vision, you're off to a false start. Better stop, do some planning inclusive of understanding the business, then restart later. Typically this means the first project was moving in the wrong direction...it's what you get when you take a shot in the dark.

What's needed is a focus on the requirements for the business and the enterprise. What issues are currently present? How do we solve them? And, develop a core strategic plan for doing so. Moreover, what about ROI and a good definition of the value?

Second, it's not SOA. Many projects are called SOA, but are really not SOA. Often I see portal development projects that are funded with the title "SOA" but have nothing to do with services, agility, reuse, or architecture; they are just portal development projects. Portals are great and can add value to the business, but they are not SOA unless it's part of a larger plan and strategy. At the end of the day, SOA is architecture.

The trick here is to shine a light on the spin that occurs during a project's sales pitch. Ask core questions as to the real value of this technology for the architecture, and not just the value of the technology. Those are very different questions.

Finally, there are not enough resources. Many SOA projects are under funded, typically because they are considered strategic and long term, with more resources directed at short term and tactical. I would rather cancel the project and start again later, than have a lot of well-intentioned people spend time making changes to the architecture that will not have any value.

What needs to be clear upfront is the amount of money that's required for SOA, both hard and soft dollars. SOA is expensive and hard to do. Thus, the enterprise needs to make sure that enough resources are allocated to SOA to make it work, otherwise don't bother.

The core issue here is misdirection. People have a tendency to forget that SOA is architecture and not technology. SOA is long-term strategic, not short-term tactical. SOA is something you do, not something you buy.

The truth is that many of these bad projects have received funding, resources have been allocated, and they are in motion. At this point, somebody has a corporate responsibility to say something and do something, including pulling the plug on the project. Typically this is not a popular move, but making the right call almost never is.

Hopefully you're not working on a SOA project that needs to have the plug pulled, but if you are, you may have to make some tough calls. ■

About the Author

David S. Linthicum (Dave) is an internationally known application integration and SOA expert. In his career Dave has assisted in the formation of many of the ideas behind modern distributed computing including Enterprise Application Integration, B2B Application Integration, and SOA, approaches and technologies in wide use today. He keynotes at most major SOA and Enterprise Architecture conferences, maintains one of the most read SOA blogs, is the host of the weekly SOA Report Podcast, and is the author of 10 books, three on integration and SOA topics.

linthicum@att.net

ALTOVA® mapforce® 2008

Drag & Drop

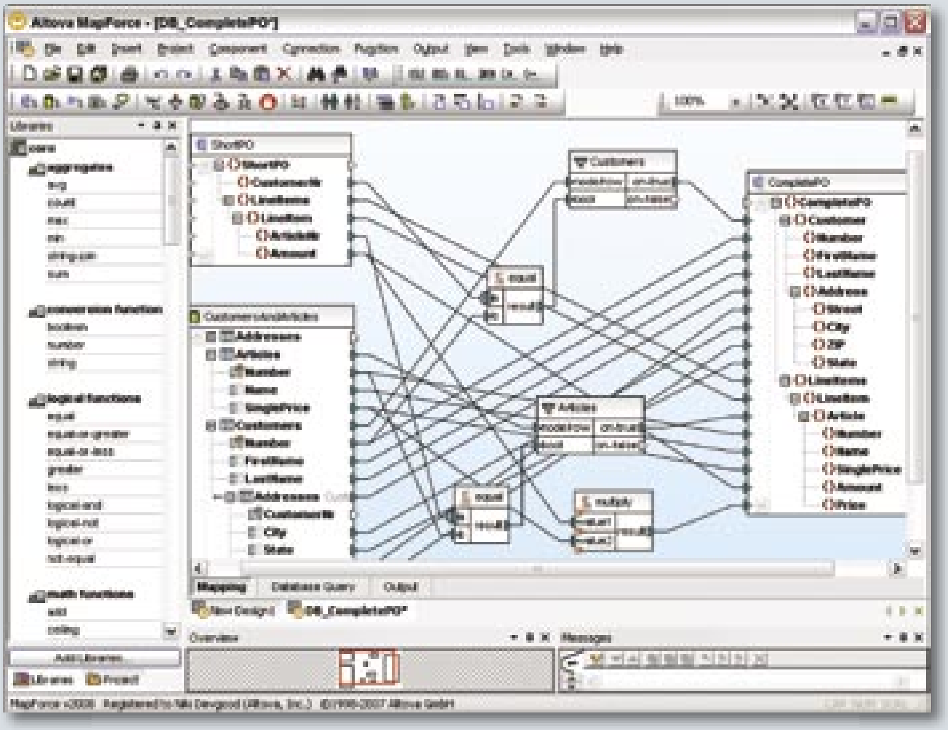
Data Conversion

Check out Altova MapForce® 2008 – the award-winning graphical data mapping tool from the creator of XMLSpy. Drag & drop to map, convert, and transform data between:

XML **Databases** **EDI** **Flat Files** **Web Services**

- Drag & drop data mapping & conversion
- Support for all major relational databases
- Database query window with SQL editor
- FlexText utility for parsing flat files
- Support for EDIFACT and X12 EDI messages
- Connecting data to Web services
- Auto-generation of XSLT 1.0 and 2.0, XQuery, Java, C#, or C++ for royalty-free use
- Drag & drop Web services creation
- Extensible function library for filtering / processing data
- Visual function builder for custom functions
- Instant data conversion & output window

Once you have defined a data mapping in MapForce, simply click the Output Window to convert data instantly. Or, generate royalty free code – and deploy it with no additional fees or deployment adaptors required. With MapForce, you can implement data integration and Web services applications without writing any code!


Altova MapForce - [DB_CompletePO*]
File Edit Insert Project Component Connection Function Output View Tools Window Help
Libraries
ShortPO
Customers
CompletePO
Articles
Mapping
Database Query
Output
Overview
Messages
Libraries
Project
MapForce v2008 ©1998-2007 Altova GmbH

ALTOVA®

Test drive MapForce for yourself – Download a free, 30-day trial at www.altova.com