SYS-CON

THE WORLD'S LEADING MAGAZINE DEDICATED TO WEB SERVICES TECHNOLOGIES

FOR ALL THE LATEST NEWS AND INFORMATION VISIT:

www.SOA.SYS-CON.com

SOA WORLD MAGAZINE

NOVEMBER 2007 / VOLUME: 7 ISSUE 11

Real-Time SOA Starts with the Messaging Bus!

_INFRASTRUCTURE LOG

_DAY 79: This is out of control! Our IT environment is rigid and inflexible. Our business needs are changing, but our environment isn't built to change with them. We can't adapt. Oh, no...I was afraid of this. We're so rigid we're stuck in time.

_*Infrastructurus prehistoricus*. I've read about this.

_DAY 80: I'm taking back control with IBM SOA solutions. Now we can align business goals with our IT. We have the hardware, software and services we need to respond to change. Strategy, planning and implementation are in tune with our specific business needs. Now we can deploy and update business processes faster and more efficiently.

_Goodbye, rigid past. Hello, flexible future.

Take the SOA business value assessment at:
IBM.COM/**TAKEBACKCONTROL**/SOA

Speaking in Tongues

WRITTEN BY **SEAN RHODY**

Recently I had a chance to do some training in France. I participated in a week of coursework with classmates from all over the world. Some were from France, Spain, Holland, Sweden, and England; others were from even further – the United States and even India. To say the least, it was an eye-opening experience and dramatic evidence that standards and interoperability are important in all aspects of our work as technologists.

Early on in our journey to a service-oriented enterprise we began to realize that for services to work, truly work, they had to be standards based, and, more important, those standards had to trump implementation, so that the services would interoperate as well.

In our training class, English was the language standard for speaking as well as writing. This made sense, as English seemed to be the one language that the class all had in common. The question was posed, for example, since the training was in France, why not have a French version of the class. This seemed a reasonable question to me as well. The answer was that while we were in France, there were many participants who did not speak French. Also, given the international nature of the audience, the question could be posed for many other languages as well. For the purposes of cost containment and the ability to be flexible regarding the content (by only having to update content in one language, not a dozen), a single standard was needed.

This lesson has very similar applications in our work. Not everyone spoke English with equal facility – some were very fluent, while some could understand what was said but had less confidence in their own speaking abilities. Similarly the issues faced by various implementation differences from vendors resulted directly in the creation of the WSI board and its interoperability profile. In our analogy, the basic profile is sort of like a test of spoken and written English that allows users to gauge their ability to communicate successfully with other users. Which is all the basic profile does. It doesn't mean that every word is understood by both parties, merely that they can understand each other. Occasionally we still had to resort to pictures, pointing, or the odd written word to be adequately understood.

I've seen this happen recently due to simple differences in WSDL between the Apache Axis toolkit and one of the commercial ESB products. While the WSDL worked on one side, Apache, it refused to function as is on the bus. Sadly, the developers responsible for this electronic tower of Babel sat right next to one another, so there was no reason at all why they couldn't have communicated with each other and shared the WSDL early enough in the process so that when the integration work was scheduled to happen, it would have gone smoothly.

The coursework did show another amazing aspect of standards and interoperability – how successful people could be when they were able to communicate effectively. During the training exercises, I'd say the average participant spoke at least three languages (I do, but only if you count Latin). Yet we were able to do the work effectively and have very intense discussions around the subject matter, because we all shared one common standard.

In the same fashion, when you consider what the standards that we have now allow us to accomplish, it's pretty remarkable. Ten years ago we had major IT initiatives designed to allow systems to speak with one another (we called it EAI). Sadly, each of those initiatives was in its own language, so at large we still had chaos and no one lingua franca, and no Rosetta stone. Now, thanks to our standards around services, we have achieved interoperability and communication without the need for proprietary solutions and can accomplish the majority of EAI functionality via our existing platforms (with perhaps a few additions such as BPM). Fortunately for all of us, we are no longer speaking in tongues.

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*

SOA WORLD MAGAZINE
www.SOA.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 (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

Mindreef®

SOAPscope server™

Whether you're an Architect, Developer or Tester, rely on the Web services and SOA tools that won't leave you stranded.

Mindreef provides an integrated set of tools at the XML software layer for SOA and Web services policy governance, testing, diagnostics and support. Available in server or workstation licensing options, Mindreef SOAPscope® products help professional developers, testers and architects working alone or on teams to understand and address SOA and Web service problems quickly and cost-effectively.

Visit our website and learn why more than 3,000 customers worldwide, including 40 of the Fortune 100, use Mindreef products to successfully build, test and maintain high-quality Web services and SOAs.

SEE IT!

Register on our website to attend a webinar on Automating Web services and SOA testing

TRY IT!

Request an evaluation copy of Mindreef SOAPscope Server or SOAPscope Workstation™

BELIEVE IT!

Start saving time and money, and improving the quality of your Web services and SOA projects

The SOA Quality Company™

www.mindreef.com

(603) 465-2204 ext. 581

SOA and Mass Data Movement via ETL

The art of preserving legacy systems

WRITTEN BY **SUREKHA DURVASULA**

On the one hand, there's extreme pressure on businesses to deliver new customer offerings and innovative business capabilities, match increased competition, and deal with new partners and providers of niche services to offer cheaper service.

On the other hand, most of us have legacy systems that prevent us from delivering to the business at the speed of business opportunities. Legacy applications house only parts of the business entity and the data is stored in proprietary structures. The data is also owned by proprietary application logic (packaged or custom-built) that represents very narrowly defined business functions where the business rules might be applicable for a single business area or line of business.

The industry is looking to SOA as a mechanism to help businesses become more agile. However, the question is how this would be achieved in an enterprise that has numerous critical path legacy systems that have the limitations mentioned above.

This article provides one possible solution to this conundrum. The architecture requires the transport of legacy data on-demand or in a scheduled manner to a common staging repository that lets the service layer apply non-legacy, non-line-of-business-specific business logic for enterprise-wide information sharing. Further, the service layer has logic that works on the base business data captured in disparate sources across the enterprise and transforms this into information.

This architecture pattern, displayed in Figure 1, offers a possible alternative to insulate the legacy systems from changes while promoting the reuse of the data collected by the legacy system. The solution also enables IT to react to the new business rules that need to be applied to one or more aspects of a business entity.

The fundamental theme encapsulated in the architecture pattern is the fact that an enterprise can leverage powerful ETL tools and processes to gather, reconcile, and finally populate enterprise repositories in an attempt to reuse information stored in multiple legacy data structures/sources. Also shown here is the fact that the consumer is unaware of how the service provider gathers the relevant data and what the sources of the data might be. The consumer's only view to the business entity and the business rules governing the business entity is via the use of an enterprise-worthy service provider interface.

Standardized ETL processes are leveraged to transport, reconcile, and transform the granular data that represents various aspects of the business entity into enterprise business information, while services are leveraged to apply enterprise-wide business rules to make business information and business functions available to the business in a consistent manner.

The pattern also demonstrates how an enterprise can extend the use of specialized or silo legacy data to service-enable all of the various legacy applications. The cleansed and reconciled data is populated into an enterprise repository that is then accessed using enterprise rules. This insulates the consumer from having to know or care about the details of invoking these legacy batch processes. The service provider offers a clean layer of indirection between the consumers and the sources of enterprise-worthy information that is locked up in legacy repositories.

– CONTINUED ON PAGE 32

About the Author

Surekha Durvasula is the Manager of the corporate Enterprise Architecture Group for Kohl's Department Stores in Wisconsin.
surekha.durvasula@kohls.com

SOA WORLD 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

SYS-CON EVENTS

Event Manager
Lauren Orsi lauren@sys-con.com

Event Associate
Sharmonique Shade sharmonique@sys-con.com

CUSTOMER RELATIONS

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

SYS-CON.COM

VP information systems
Bruno Y. Decaudin bruno@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: $69.99/yr (12 issues)
Canada/Mexico: $89.99/yr
All other countries: $99.99/yr
(U.S. Banks or Money Orders)

Worldwide Newsstand Distribution:
Curtis Circulation Company, New Milford, NJ

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

SOA for SMB's?

small budget
BIG QUESTIONS
mission-critical services
MAJOR PRESSURE
short timeline
reduced staff

JaxView

A flexible Web Services management solution for small and medium-sized businesses that need the following capabilities:

- Monitoring of Web services
- XML gateway and authentication
- Runtime SOA policy enforcement
- Automated Web service discovery
- Runtime message modification and routing

Download a free trial at www.managedmethods.com

JaxView
versatile, scalable, affordable

Managed Methods

Real-Time SOA Starts with the Messaging Bus!

The mediator of all component interactions

WRITTEN BY **GERARDO PARDO-CASTELLOTE**

Service Oriented Architectures are increasingly being used to implement high-performance and real-time systems. Traditional systems operate in "human real-time," where human patience is the limit. Increasingly, however, systems operate in "computer real-time," where the only limits are imposed by the operational speed of the computers and networks.

For example, next-generation Air Traffic Management systems are being developed to accommodate the huge increase in air traffic and link the operational capabilities of agencies such as the Federal Aviation Authority, the Department of Defense (DOD) and the Department of Homeland Security (DHS). These systems require higher information bandwidth (to track more aircraft or more complex "free-flight" trajectories) as well as much lower latencies or delays on the information (to detect flight abnormalities quickly). Similar demands are being made in healthcare, SCADA, network monitoring, energy distribution, transportation, and other critical infrastructure systems.

Best-of-Breed SOA Components

Demanding real-time applications require best-of-breed service-oriented foundational components. There are three kinds of foundational components in a SOA system: A messaging fabric/bus, information transformation/processing engines, and persistence/storage services (see Figure 1). Often these components are integrated into an Enterprise Service Bus (ESB) and hosted in a J2EE Application Server.

Of these foundational components, the Messaging Fabric/Bus is the most critical, since it mediates all interactions between components.

Low-performance SOA systems may use HTTP as the "messaging fabric/bus" to exchange messages between components. This approach is only suitable for non-demanding applications: HTTP isn't reliable, has limited bandwidth, introduces very high latencies, and can't buffer and queue messages and deliver them to systems that are either temporarily unavailable or join at a later time.

The solution is to deploy a high-performance messaging middleware such as RTI Data-Distribution Service, IBM WebSphere MQ, TIBCO, or SonicMQ. These middleware platforms have been developed with scalability and performance in mind. However, they each employ a different architecture optimized for different application scenarios.

Why Does Messaging Performance Matter?

The requirements and expectations of computer-speed real-time far exceed traditional human-speed real-time. Whereas in systems with a human in the loop, real-time meant that the information was available anywhere from fractions of a second to few seconds in the computer-to-computer world, real-time means decisions should be made in milliseconds or even microseconds.

Computer real-time puts more stringent requirements on the messaging infrastructure: Each processing and storage component must get hundreds of thousands of messages/events per second with microsecond or at worst millisecond latencies. This means that the

messaging middleware must be able to deliver millions of messages a second system-wide.

And the capacity of the messaging fabric must be able to scale with the capacity of the underlying hardware and not impose any limits beyond those of the underlying hardware infrastructure (CPU speed, cores, speed, and bandwidth of the network) itself. As the CPU and network speeds increase those systems able to take advantage of what the hardware provides will deliver a competitive advantage. In an automated trading system, for instance, the critical metric is not the absolute time it takes to make a decision, but rather whether a decision is taken and the trade executed before competitive trades occur. The same is true in a combat management system.

One final aspect of computer real-time SOA systems is their "inverted performance-load utility curve." This means that the ability to respond in a timely manner becomes more important when the system is experiencing a high load. In a normal utility curve, such as in human real-time systems, degraded performance is acceptable under an increased load. This is because human expectations and patience adjust based on the circumstances (e.g., they understand that on a peak holiday period they may endure longer hold times when calling to make a flight reservation). In contrast, computer-speed real-time systems often have the opposite demands. It is precisely at the moments of high load when the "most critical action" is taking place and it is then when it is most critical to deliver top performance (e.g., it is precisely when market action is heavy that trading decisions must be made quickly).

The differences between human-speed real-time systems and computer-speed real-time systems are summarized in Table 1.

Aspect	Human-Speed Real-Time	Computer-Speed Real-Time
Typical Response Times	Fractions of second	Fractions of millisecond
Typical Message and Event Loads	Dozens to hundreds per second per client Thousands per second system-wide	Tens of thousands to millions per second per client Tens to hundreds of millions per second system-wide
Evolution of Performance Requirements Over Time	Constant. Tied to human response times	Increasing. Tied to what the hardware CPU and network infrastructure can provide
Performance-Load Utility Curve	Normal. Degraded performance more acceptable under increased load	Inverted. Performance becomes more important under increased load

Table 1

Selecting Messaging Middleware in SOA Systems

Messaging middleware is the key enabler of real-time SOA. However, there are many options. How can you choose the best messaging middleware for a particular real-time SOA system? Five areas distinguish messaging middleware: architecture, quality of service (QoS) control and filters, performance-boosting technologies, real-time determinism, and metrics.

Architecture

The four basic architectures employed by messaging middleware are: centralized (hub-and-spoke), clustered, federated, and peer-to-peer. (see Figure 2)

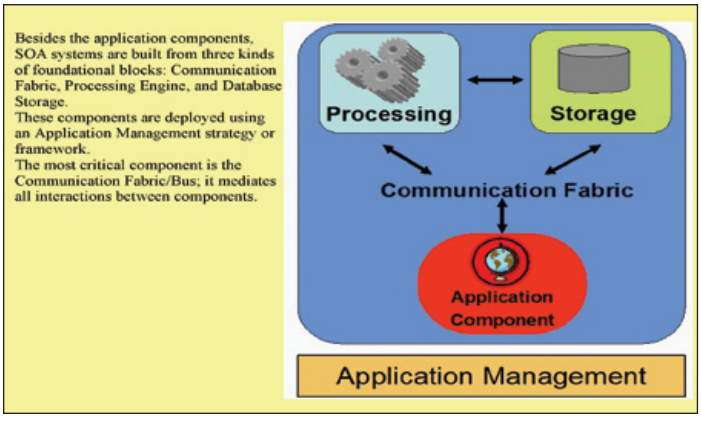


Figure 1:

A centralized (hub-and-spoke) architecture routes every message though a single server that implements the message "service," contains all the message queues, and brokers every message.
A clustered architecture uses a collection of servers and assigns to each responsibility for some of the messages (like ownership of some of the message queues or topics). Each message is relayed by a server but not all messages use the same server.

A federated architecture also uses a collection of servers, but it uses them as a "resource pool" where queues may appear in multiple servers, and messages may be brokered by one or more servers.

A peer-to-peer architecture doesn't employ any brokers in the critical path. Messages are routed directly from the sender to the receiver.

Each has strengths and weaknesses. Centralized is easiest to administer and can provide stronger transactional semantics but suffers from poor performance, reduced tolerance to faults, and doesn't scale. Clustered is more scalable than centralized but also has reduced fault tolerance and can only offer good performance in a grid environment with all the clients co-located close to the grid. Federated is more scalable, but suffers from higher latency and jitter as each message is brokered by at least two servers. P2P offers the best scalability, performance, lowest jitter, and highest resilience, but is difficult for vendors to implement and offers limited transactional support.

As demands become more real-time, the need for performance, predictability, and balance tips the scale towards P2P architecture. That's why, for example, demanding networks like Voice over IP and Video over IP (like Skype) use peer-to-peer designs.

Quality of Service Control & Filters

QoS control is critical to deliver timely data with low latency and high throughput. CPU, memory, and network bandwidth resources must be shared among all the traffic. However, not all traffic requires the same bandwidth or has the same urgency or level or criticality. Without QoS control, the application has no way to differentiate different traffic classes and their corresponding constraints. As a consequence, the middleware can't make intelligent decisions, prioritize traffic, or ultimately meet the application requirements.

For example, in a traditional network running TCP (Transmission Control Protocol), a small and urgent alarm message may be queued on a TCP connection behind a large file transfer. Since TCP delivers every byte, regardless of its source, reliably and in order, the alarm won't be received until the whole file has been successfully transferred. Similarly, imagine a situation were live video is being transmitted across a wide area network (WAN). Congestion may cause some images to be dropped; without QoS control, the middleware may insist on retrying dropped old images rather than bypassing them in favor of the latest images. Rather than miss a few unimportant frames, the image freezes until the retries succeed. So,

QoS Control	Purpose Example	Use Cases
Reliability	Let the application decide whether messages should be confirmed and retried when missed, or else sent as best efforts.	Send live voice or video data. Send sensor data (e.g., radar tracks), traffic readings, CPU/network statistics and readings.
Latency Budget	Specify the relative importance of different messages and the maximum acceptable delay between the time the message is sent and the time it's delivered to the reader(s).	Prioritize real-time flows like live audio over traffic that may be buffered (e.g., video replay). Prioritize critical control information (e.g., live radar tracks) over non-time critical information such as aircraft schedule changes.
Flow Control	Control how much load and bandwidth a particular sender can inject into the network. Control the peak load, average load, and size of a burst.	Avoid a single source from overwhelming the network. Prevent large low-urgency data (e.g., file downloads) from compromising the performance of critical data (e.g., alarms and critical news updates). Provide dedicated bandwidth to the most critical data.
Lifespan	Control how long the data must be kept by the middleware to be delivered to readers. Old data may be of little value and delivering it wastes bandwidth and gets in the way of the more recent data.	Prevent data that loses value with age (e.g., old stock values, old news, old sensor readings) from using valuable system resources, while ensuring that needed historic information is kept (e.g., transaction records).
History	Control how many related messages (e.g., successive updates to a stock value or successive readings of a sensor) must be maintained by the middleware and delivered to readers.	Prevent a rapidly changing source from using a lot of resources and starving other less-active sources. For example, an application-monitoring activity on a network may see many more events originating from a heavily loaded server than from a lightly used one. Some applications may only be interested in the last 100 events for each server regardless of the time interval when they occurred.
Transport Priority	Controls the traffic class used for the underlying network transport.	Allow exploiting the differential service capabilities of the network infrastructure, such as the traffic class and flow label fields in the IPv6 headers. Configure the network infrastructure to prioritize certain messages ahead of others.

Table 2

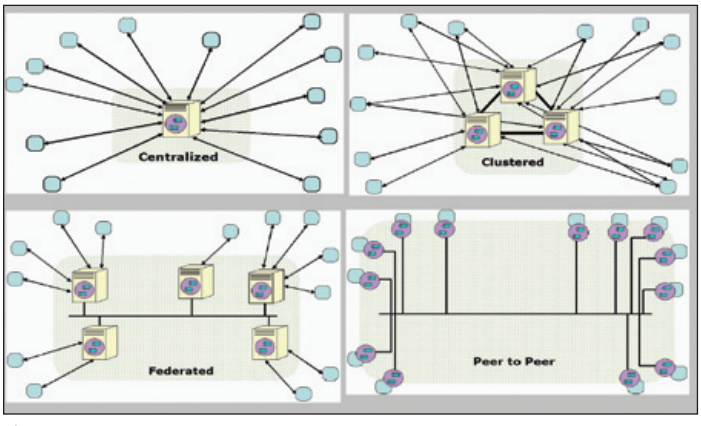


Figure 2:

even though the hardware can deliver acceptable performance, the overall system performance may degrade to the point where it fails its mission.

To achieve computer-speed real-time performance, a messaging middleware must offer at least the following QoS controls: reliability, latency budget, flow control, lifespan, history, and transport priority.

Filtering is critical for system scalability and performance. Middleware that supports filtering can deliver to each component only the messages that are of interest, saving critical network bandwidth and CPU. Ultimately this means that the system can process far more information. For example, an Event Processing Engine might be looking for events that concern network activity involving certain computers or networks, or belonging to certain protocols. Rather than delivering all the network events to the engine and having the engine filter out the uninteresting data, the middleware can pre-filter and only send data that pass the filter, saving network bandwidth and CPU in the Event Processing Engine.

There are two main kinds of filters: time-based and content-based. Time-based filters limit the information frequency, such as sending at most 10 updates a second. Time-based filtering is useful for applications, such as human-oriented dashboards, that just need to get trending and summary information. Content-based filtering reduces the information based on its contents.

For example, a visual display application may only be interested in aircraft positions when the craft is close to an airport. A content-based filter could screen out all the planes that aren't close to the airport.

Performance Technologies

No matter how well architected, a car can't have good performance without an engine that delivers good torque. Similarly, there are certain features and technologies that messaging middleware needs to deliver top performance. These stand out: Multicast, message batching, message fragmentation, asynchronous (non-blocking) writes, and zero-copy access to the transport.

Real-Time Determinism

Real-time is not just about being fast. To be reliably real-time, the system must also be reliably fast. Real-time determinism determines how predictable the system will be for each operation. A deterministic real-time system will do every operation in the same amount of time every time.

Ironically, achieving real-time operation isn't so much about what you do; it's what you don't do that counts. Real-time requires predictability and consistency in processor use, resource allocation, and management. Allocating memory, wait loops, disabling interrupts, or any of hundreds of other common programming operations can introduce unpredictable behavior. To be predictable, an operation must behave the same way and take the same time every time. For an operation composed of many steps, each and every step must be predictable, since sources of unpredictability compound; if any single step is unpredictable, the whole chain becomes unpredictable.

Since the underlying hardware components are usually very predictable the sources of non-determinism in most systems can be traced to the operating system, middleware, and application logic/code. Real-time operating systems are well-known technologies. Applications must be designed to deliver predictable results. However, in many systems, the middleware is the key to real-time operation.

In practice, this means the messaging middleware must:

- Use asynchronous operations rather than blocking operations. Provide multiple threads to exploit opportunities for concurrency and multi-core architectures.
- Use multiple threads so that competing tasks can be concurrent. Carefully pick the architecture and priorities to avoid thread contention and include watchdogs and timeouts to monitor and maintain application liveliness.
- Execute on a predictable infrastructure (real-time operating system, real-time JVM, real-time middleware components).
- Carefully control resources. Control the use of dynamic memory

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®

Performance & Scalability Boost	Description	Why It Matters
Multicast	Internet technology that allows a single UDP message to be delivered to many receivers.	Provides the most efficient way to send messages to multiple receivers. Reduces bandwidth, reduces overhead on the sender, and minimizes latency and jitter.
Message Batching	Middleware technology that combines multiple messages into a single unit. (Also known as message aggregation.)	Greatly increases the throughput for small messages. Reduces bandwidth and processor utilization for small messages.
Message Fragmentation	Middleware technology that breaks a large message into smaller units, delivers them separately, and then reassembles them prior to deliverance to the application.	Enables multicast use for larger (greater than 64KB) messages. Prevents "Head of Line" blocking where a high-priority message is queued behind a large message. Reduces jitter. Provides better performance in less reliable networks (wireless environments or WANs).
Asynchronous Writes	Middleware technology that allows a write operation to be processed by a separate thread and not block the application thread that performed the write.	Decouples sender and receiver, providing more predictable performance for the writer and reducing latency jitter. Allows multiple write operations to be performed concurrently over multiple channels, batched, or optimized in other ways.
Zero Copy	Operating system network-stack technology that allows an application to put and get data from the network buffers "by reference," without performing extra copy operations.	Increases performance for large messages, resulting in higher throughput and lower latency. Reduces CPU consumption on both sender and receiver. Makes performance scale better with message size.

Table 3

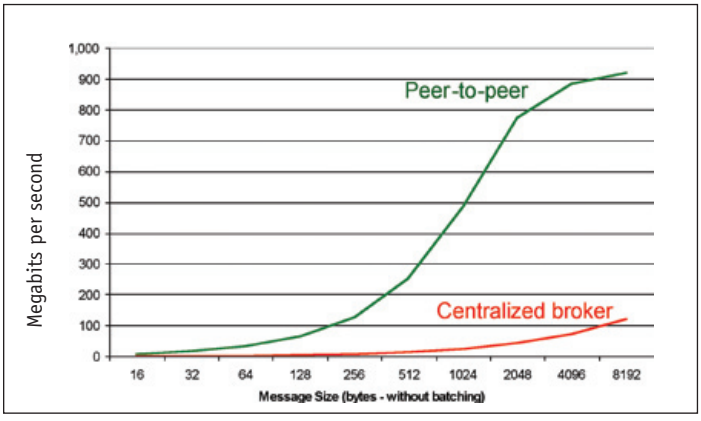


Figure 3:Throughput

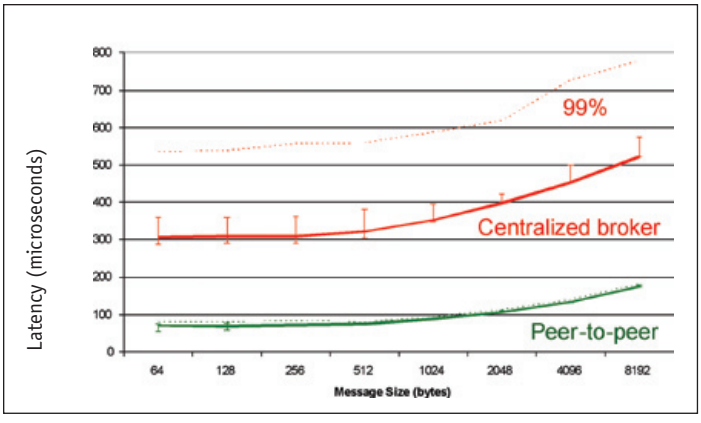


Figure 4: Latency and Jitter

and limit allocation to activities outside the critical path.

- Use in-memory caching rather than direct-disk operations for any storage access.
- Provide an "in-process" mechanism (e.g., loading DLLs or Java classes) for any application-programmable extensions to the middleware.

Metrics

Ultimately good metrics are the only way to determine what the middleware can deliver. Unfortunately, characterizing messaging middleware performance is far more complex than it seems at first glance.

Most vendors provide some indication of messaging throughput measured in terms of messages or bytes per second. A few offer latency measurements (end-to-end delay from sender to receiver). Almost none provide scalability and jitter metrics (how the performance changes as the system grows, as the load increases, or how much variation there is from message to message). Furthermore, the results are highly dependent on the computer and network hardware as well as the testing scenario.

For example, a vendor may provide both latency and throughput measurements and say that the product can deliver messages in less than 0.5 milliseconds, and deliver over 1,000,000 messages a second, but not indicate whether an application can accomplish both simultaneously. Moreover, it's not clear whether the latency represents the best it can be under ideal circumstances, an average, a guarantee that most messages (e.g., 99.99%) will be deliver

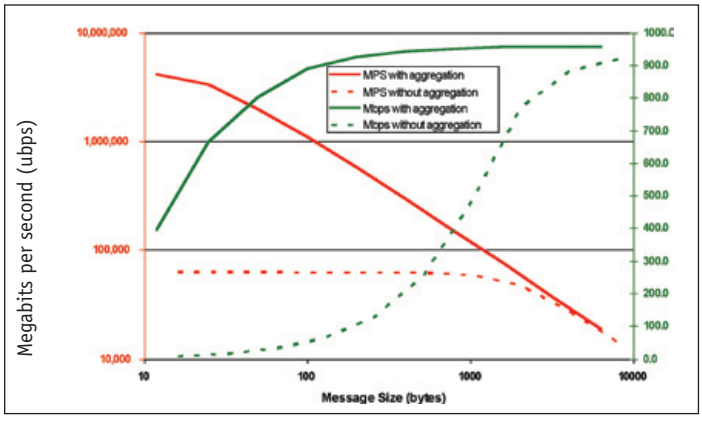


Figure 5:

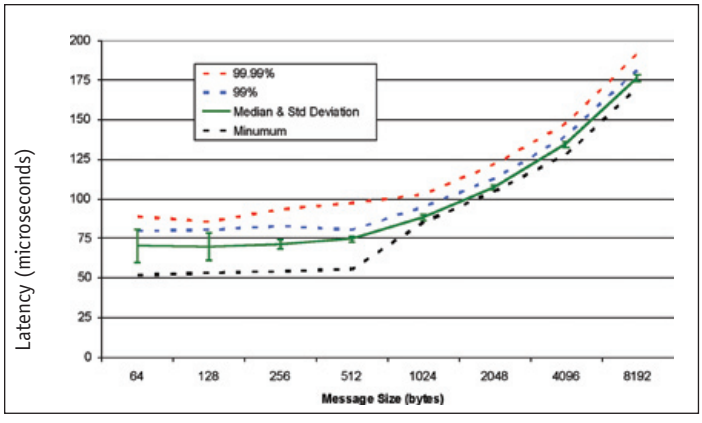
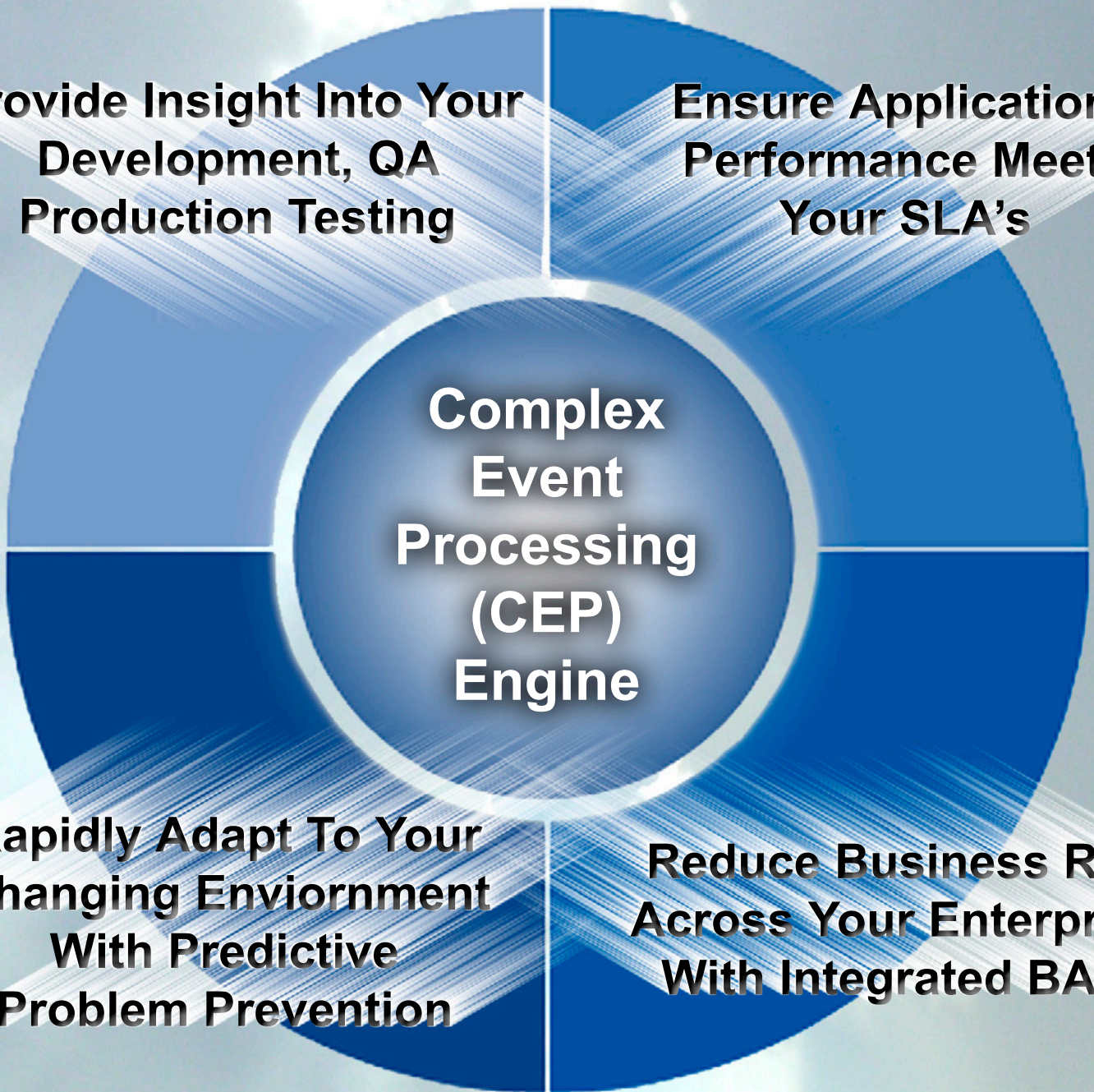


Figure 6:

We Shed the Light On SOA

Solutions For Complex Application Performance Management



NASTEL™

We work with some of the world's largest banks, financial services, retail chains and insurance companies. IT managers in these companies have to comply with stringent service level agreements (SLAs) and government-mandated regulations and reporting requirements. We provide the most flexible, powerful mechanisms to ensure the performance of the complex applications that drive business by quickly finding and fixing problems to meet business requirements.

Stop by our booth at SOAWorld 2007 West from November 12-13th at the Grand Hyatt in San Francisco to learn how your business can benefit from using AutoPilot M6™. For more information, visit http://www.nastel.com.

in that time or less, or some other metric. Message size and the type of content (opaque bytes, strings, or complex type) have a big impact. For instance, "messages per second" may have little meaning if it represents unrealistic four-byte messages. Bytes per second aren't meaningful if the type translation (marshaling) isn't considered. For example, inefficient data marshaling that sends raw XML strings would send many bytes per second. However, efficient marshaling that used compressed XML or a binary format would deliver the actual message much faster. The list goes on and on.

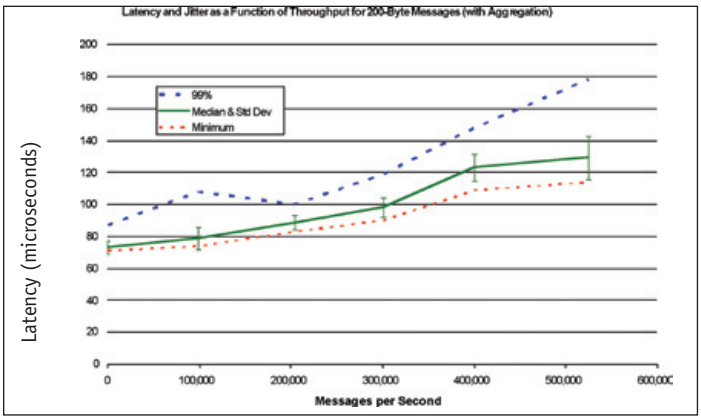


Figure 7:

Metric/Aspect	Description	Why It Matters
Throughput versus Message Size	Throughput measures the number of messages per second or the number of bytes per second that can be sent from a sender to one or more receivers. Message size has a big impact on throughput.	Determines the information processing "capacity" on the system. The more information that can be delivered to the engines, stored, etc. the more intelligent and sophisticated the decisions can be.
Throughput versus CPU	Measures the CPU cost of achieving higher throughput.	Characterizes the efficiency of the middleware. The more efficient, the lower the CPU consumption for a given level of throughput. This saves hardware costs and leaves more CPU for the application logic, increasing the overall information processing capacity.
With and Without Batching	Compares the above throughput metrics when batching is in use.	Characterizes the efficiency of the middleware for both small and large messages. Small messages are most affected by batching performance.
Best Efforts versus Reliable	Measures the above throughput metrics for the different communication modes: reliable/guaranteed and best efforts.	Characterizes the cost of achieving reliable/guaranteed delivery. Characterizes the suitability of the middleware for "fire and forget" applications such as live voice or video.
Latency versus Message Size	Latency measures the time elapsed from the moment the application requests a message be sent till the moment the receiving application gets the message. Message size has a big impact on latency.	Characterizes the suitability of the middleware for time-critical applications such as "active control" systems, stock trading, combat management systems, etc.
Latency versus Throughput	Measures the achievable latency for each value of throughput.	Latency and throughput interact in conflicting directions. The same techniques that increase throughput (message batching, asynchronous messaging) increase latency. Characterizes the values of latency that can be achieved in more "realistic" scenarios where the system is under load.
Latency & Jitter statistics: 50%, 90%, 99%, 99.99%	Measures the maximum delays required to statistically guarantee the arrival of a large percentage of the messages. For example, a latency of one millisecond at 99% indicates that 99% of the messages take one millisecond or less.	Characterizes the jitter and suitability of the system for time-critical applications. Allows assessment of the likelihood that message delays could be unacceptable. Scopes potentially fatal results or financial losses associated with messages being delayed

Table 4

Since standardized benchmarks that measure real-time middleware performance don't exist, the only options are: (a) rely on the vendor's numbers, (b) develop your own benchmarks, and (c) judge the middleware performance by the kinds of applications that use it. The first is subject to ambiguities, the second is long and costly, and the last is subjective.

With all those caveats, understanding the following metrics will help assess middleware performance and should minimally be required from any vendor that targets real-time SOA systems: throughput versus message size, throughput versus CPU use, message batching performance, and best effort versus reliable. Latency versus message size, latency versus throughput. Latency & jitter statistics: Mean, 50%, 90%, 99%, 99.99% for small, medium and large messages.

For example, Figures 3-7 illustrate the throughput and latency graphs that compare two different middleware architectures: a centralized broker design (JMS) and a peer-to-peer network (DDS). For illustration purposes, here are more detailed metrics from the RTI Data Distribution Service obtained on a 2GHz dual-AMD Opteron PC running Red Hat Enterprise Linux 4.0:

Summary

To win, a racecar must integrate the best components. Similarly, performance-critical SOA systems must be built using best-of-breed messaging, event processing engines, and databases. Among these SOA foundational components, the messaging middleware plays the most critical part, since it mediates the interactions between all other components.

How can you identify the best middleware?

First, consider the architecture. Choose an architecture that fits the likely demands of your system. No system will perform well with an inappropriate architecture.

Second, match the level of quality of service control to your needs. Good control over what the network is doing and which data is delivered greatly affects the efficiency of all the components. Third, look for middleware that includes the performance-boosting technologies your application needs. Multicast, for instance, can greatly increase the "fan out" of your network at the same load level.

Fourth, if your system performance must be reliable under many conditions, look for a design than can run at least the key components in a deterministic real-time environment. There is no other way to guarantee reliable response.

Finally, with knowledge of the caveats, choose a middleware that delivers the metrics that matter most to your application. Understand each of these factors and you'll be a step closer to building your top-performing SOA system.

About the Author

Gerardo Pardo-Castellote, PhD, is chief technology officer of Real-Time Innovations Inc.

gerardo.pardo@rti.com

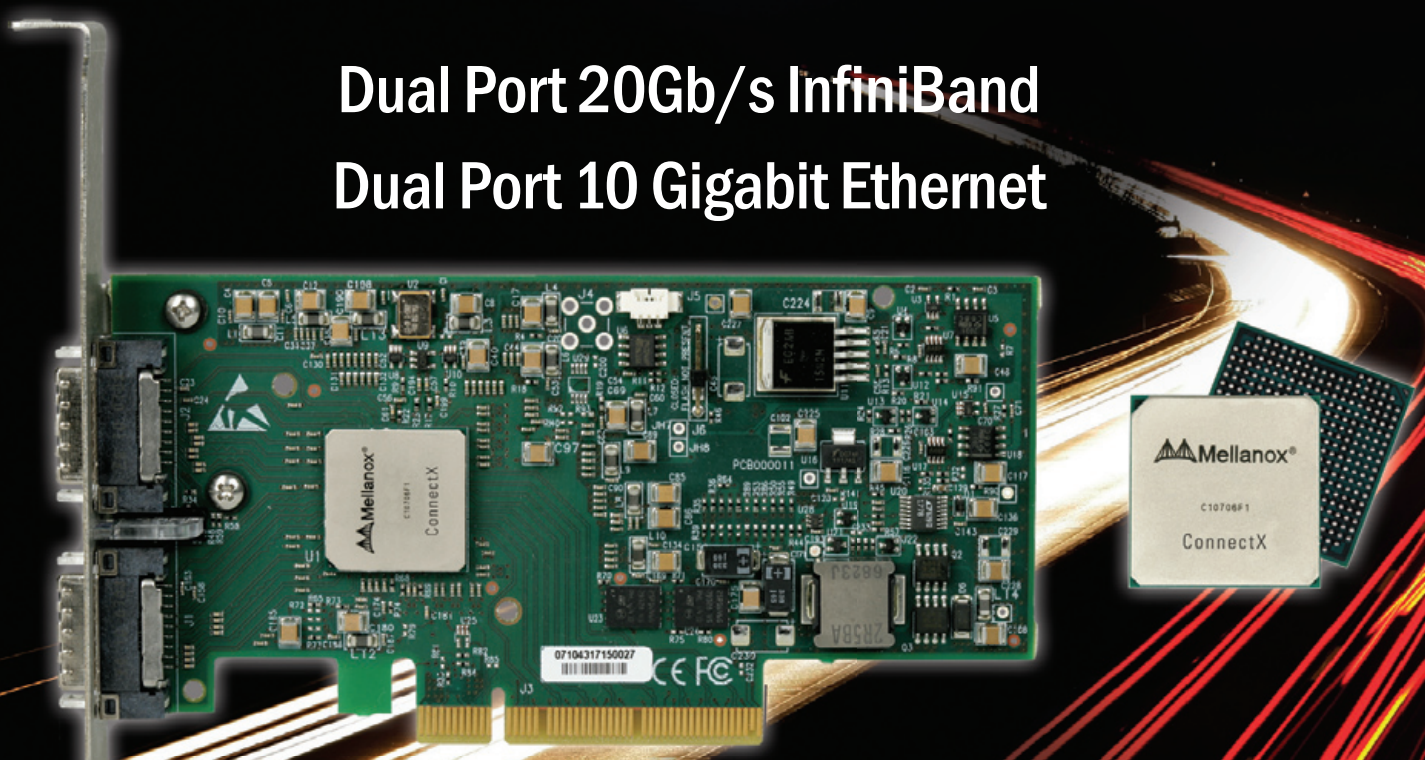
ConnectX™

Make Data Center Connectivity Your Competitive Advantage

Hardware I/O Virtualization for Maximum ROI

Dual Port 20Gb/s InfiniBand

Dual Port 10 Gigabit Ethernet



Mellanox®
TECHNOLOGIES
www.mellanox.com

2900 Stender Way, Santa Clara, CA 95054, Tel: 408.970.3400, Fax: 408.970.3403, email: info@mellanox.com

When Drug Discovery Needs Data Fast!

Information is key to drug discovery

WRITTEN BY DAN ENG

As the demand for new medicines grows, so does the need for better information to manage and execute the R&D processes. There is huge pressure to make informed decisions, especially during the project's early stages when the risk is high and before downstream costs are added.

Pfizer spends billions on research projects annually. At Pfizer Global R&D where the company's drug discovery takes place, research scientists and managers require vast amounts of up-to-the-minute information on lab results, submission status, and project schedules to move new research forward quickly. Management must constantly analyze the entire portfolio of new medicines in discovery to look for opportunities, trends, and areas where attention is needed. Researchers and managers strive to bring together the best in ideas, practices, policies as well as the use of information.

At Pfizer's Research Informatics Division within Global Research and Development, we seek to provide the best information possible to our R&D customers. Meeting this mission requires constant innovation. Over the past several years, we have faced a number of challenges, causing us to evolve our information delivery methods and technologies significantly. These include a new approach to real-time data integration, such as using SOA data services that lets us build new solutions more rapidly and in alignment with our SOA strategies.

Data Integration Is a Critical Requirement

At Pfizer R&D, the information required for executing and managing projects is drawn from many sources, including laboratory research, historical records, clinical trials, and business intelligence. The data is complex, diverse, and spreads across the company in various technology and application silos.

Through innovative use of analytics, reporting, and portal technology, we have made great strides toward improving how this information is presented internally. However, data integration remains the biggest challenge in effectively providing information to our researchers and managers.

Why is this critical? To properly assess a portfolio of discovery projects, Pfizer managers must pull data from sources such as packaged applications, historical data from data warehouses, document repositories, and custom systems. Each source has its own access mechanisms, syntax, and security. Few are structured properly for

consumption, let alone reuse. These combined factors slow down new application development projects.

Time Is of the Essence!

To move new research forward as quickly as possible, our research scientists and managers must have critical up-to-date information from across our wide array of source systems. If information is only refreshed monthly, then necessary course corrections are typically delayed by several weeks. A few weeks may not sound like a lot, but on a 24-month project, these weeks can easily add up to six months or more.

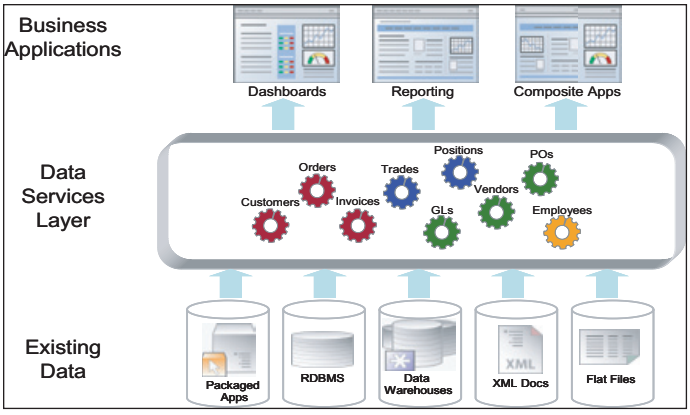


Figure 1: Data services layer. *Courtesy of Composite Software*

For new IT projects, time is of the essence. Business agility requires IT agility. Pfizer's researchers and managers, like their business user counterparts, constantly make new demands on IT for new information systems to help the business perform more efficiently, effectively, and competitively. This means we must build new systems quickly. Rapid application development (RAD) techniques are highly desired. In fact, we continuously evaluate our Enterprise Development Life Cycle (EDLC) processes with a primary objective of reducing time-to-solution with faster responses to business needs.

SOA-Compliance Is an Important Requirement

Our SOA adoption has accelerated in recent years. Specifically, we use SOA approaches to increase reuse of existing components, save development time, and cut costs. So, we strive to use SOA methods and technologies whenever possible.

With respect to SOA and data integration, we've found that SOA helps break down silo-type data gathering and integration processes by standardizing how data is promoted and reused. The ability to virtualize and abstract via data services helps groups to easily understand and consume data confidently, reliably, and quickly without having to hunt for these sources or rely on manual processes for gathering and integrating them.

Old Extract & Mart-based Approaches Can't Meet New Requirements

Traditionally, Pfizer has used three approaches to data integration. The first is custom coding directly between sources and consuming applications. This works well for our simple integration problems where there are one or two defined sources and little transformation is needed. But as additional sources are added, and complex data structures are ever changing, this delivery approach has severe limitations.

Second, we've used replicated file extracts as a way to integrate data. File extracts handle data silos more efficiently than custom coding. For example, application teams that need data receive periodic file extracts from the application teams that manage the source data applications. This arms-length batch approach minimizes the impact on source systems and is useful for daily transaction summaries, shared reference data, etc. However, data integration beyond simple access – abstraction, transformation, federation, and more – requires extra work by the consuming team. This method proliferates replicated data without any controls on quality, security, and scalability.

Extract, Transform, and Load (ETL) with data marts or warehouses is our third approach to data integration. This kind of physical data replication has several advantages in terms of rationalizing and combining heterogeneous data from multiple sources. For large-scale multi-dimensional analysis, we find data warehouses are effective solutions given their ability to support the large volumes and significant schema transformations typically required. To date, this has been the data integration approach of choice for our medium and large-scale data integration projects.

Unfortunately, these three approaches may not be entirely effective with our customers. Because our customers must make decisions based on near real-time data, they often can't afford the extra development time required for building and testing custom coding, file extracts, and data marts. Forcing our business users to wait extra months for new solutions to be developed has a huge impact on how quickly we get new drugs to market.

Further, typical data mart/replication architectures don't easily fit into our new SOA strategy. New data integration projects must be SOA-enabled from the start, so they can deliver value moving forward.

Given the accelerating business demand for new systems from the R&D groups we support, my team decided to find a new approach to data integration that lets us build additional real-time solutions more rapidly, and in alignment with our SOA strategies, while avoiding the replication downside.

To do this, we launched a project with the goal of identifying and adopting a new approach to data integration that meets the following criteria:

1. Complex data integration across multiple heterogeneous sources without unnecessary data replication,
2. Real-time information delivery,
3. Rapid application development, and
4. SOA-compliance.

SOA Data Services Approach Selected

After reviewing several new alternative approaches, we identified SOA data services as the best one for meeting our criteria.

Data services are a form of Web Service optimized for real-time data integration. Data services virtualize data to decouple physical and logical locations and therefore avoid unnecessary data replication. Data services abstract complex data structures and syntax. Data services federate disparate data into useful composites. Data services also support data integration across both SOA and non-SOA applications.

Architecturally, data services combine to form a middle layer of reusable services, or a data services layer, decoupled from both the underlying source-data layer as well as the consuming solutions layer. This provides the flexibility required to deal with each layer in the most effective manner, as well as the agility to work quickly

across layers such as applications, schemas, or underlying data sources change (see Figure 1).

Beyond providing complex multi-source data integration, data services meet our other criteria as well. Because data services are on-demand, they meet our requirement for real-time information delivery. By not replicating data, data services eliminate the time required for building and testing marts. Further, data services can be automatically generated directly from our data models and so don't require coding. Data services, due to abstraction, can often be reused across projects. Finally, data services, because of their architecture, XML support capabilities, and standards compliance, are inherently SOA-compliant.

Data Services Infrastructure Technology Selected

Once we chose a SOA data services approach, we searched for a data services infrastructure provider that offered development tools and an appropriate run-time environment. We selected Composite Software. With more than 20 projects running in various Pfizer divisions and a Composite Center of Excellence at our headquarters, Composite was a proven vendor at Pfizer and its best-of-breed offerings met our search criteria.

Now our overall data integration capabilities include data virtualization, data abstraction, and data federation across both SOA and non-SOA environments. Delivered via Composite's Information Server, the solution supports both our design and run-time requirements. At build time, we have an easy-to-use data modeler and code generator to abstract our data in the form of relational views for reporting and other uses and/or Web data services for SOA initiatives. Its high-performance query engine securely accesses, federates, and delivers the diverse distributed data to our consuming solutions in real-time.

The Proof Was in the Portal

With our data services strategy and data integration toolset in hand, our next task was to do a pilot project. We wanted to see if we could successfully complete the project, and if we could complete it much faster while complying with SOA principles.

For our pilot, we selected the Drug Discovery Portfolio portal. This project easily met our evaluation criteria.

ETL / Data Mart Approach	Data Services Approach
Define portal and portlet data requirements	Define portal and portlet data requirements
Model data schema	Model data schema
Code the ETL scripts	Generate data services automatically from schema
Set up the data mart infrastructure	
Run the ETL scripts to load the mart	
Unit test ETL scripts and mart	Unit test data services
Modify schema and ETL scripts	Modify schema, regenerate data services
Reload mart using revised ETL scripts	
Build portal and portlets	Build portal and portlets
Bind portlets to mart	Bind portlets to data services
System test portal	System test portal
Modify schema and ETL scripts	Modify schema, regenerate data services
Modify portlets	Modify portlets
Reload mart using revised ETL scripts	
Rebind revised portlets to revised mart	Rebind revised portlets to revised data services
Retest	Retest
Go live	Go live

Table 1: ETL/Data mart approach vs data services approach

Business Requirements

Senior management, project team leaders, business analysts, and research scientists across Pfizer's R&D and commercial business units need to continuously evaluate our portfolio of discovery projects and drugs in development. This analysis includes how these projects fit into Pfizer's overall strategic portfolio as well as how each will be impacted by costs, market conditions and available resources. A complete picture of each particular project, as well as an overview of all the projects, is needed for major business decisions to be based on all relevant factors. Real-time access to this information is critical, so Pfizer can rapidly react to unforeseen events intelligently.

User Interface Requirements

We selected a Web portal as the user interface because this provides the most flexible and accessible solution for our wide range of information users. This means existing data has to be delivered in the form of Web data services for our portal developers and our portal toolset to consume easily.

Data Integration Requirements

Key data to be delivered includes both key metrics and details such as project costs, resources, timelines and ROI calculations, to name a few. This diverse data needs to be integrated from a wide variety of source applications from across various Pfizer groups. This diversity of source system data structures enabled us to evaluate and thoroughly test Composite's data connector and transform capabilities during the pilot project. We also thoroughly tested Composite's high-performance query algorithms through the dynamic nature of the sources and the need for real-time delivery. Because many teams from across the globe needed to be involved to provide access to the right data, we added ease-of-use to our RAD evaluation criteria.

Pilot Benchmark: The Data Mart Approach

To compare the relative and absolute strengths and weaknesses of the new data services approach and the Information Server versus our traditional approach, we invested in a small benchmark of the "old way." Benchmarking the functional and technical specifications lets us compare end solution delivery. Benchmarking the development process lets us compare time-to-solution and development costs.

Functional and Technical Specification

We already knew we could use our ETL/data mart tools to successfully combine the data required into a mart. Unfortunately, putting the relational data into a mart was only half the job. We still needed to get this data out of the mart and into the portal in the form of a Web Service. We found this requires manual coding and an additional toolset. What's more, to achieve the real-time delivery requirement, we found we needed to achieve unrealistic refresh rates using highly complex change data capture techniques.

Development Process

In a side-by-side comparison, Table 1 represents the steps used in an ETL versus a data services approach.

Problems with the Data Mart Approach

The ETL/data mart approach was not ideal for this project for the following reasons:

- We could only come close to meeting the real-time integration requirements if we used advanced change data capture and fre-

Altova® UModel® 2008 – The starting point for successful software development.

ALTOVA® umodel® 2008

Visualize works of software art

Draw on UModel® 2008, and picture better programs based on UML™.

Spied in UModel 2008:

- Support for all 13 UML 2.1.1 diagrams plus a unique XML Schema diagram
- Code engineering for Java, C#, and now VB.NET
- Visual Studio and Eclipse integration with real-time model synchronization

Altova® UModel 2008, the developing force in the software design space, is the simple, cost-effective way to draw on UML. Use it to create or interpret software architectures. Visually design application models and generate Java, C#, or VB.NET code and project documentation. Or, de-code existing programs into clear, accurate UML 2 diagrams. With all 13 diagram types, interoperability via XMI 2.1, an artful user interface, and more, UModel makes visual software design practical for any project. Take the mystery out of UML!

Download UModel® 2008 today:
www.altova.com

UModel is also available as part of the value-packed Altova MissionKit™ software bundle.

ALTOVA®

quent refresh features.

- We found that the data mart was physically instantiated in a relational form. Yet, our portal developers wanted the data in the form of WSDL Web Services that are easier for the portal to consume.
- Sequential development such as building the ETL scripts, the mart, the delivery scripts, and then the portal application stretched the elapsed time thereby pushing out business benefits and adding costs.
- ETL and Web Service scripting were slow manual development processes.
- Scheduling the setup of the data mart infrastructure required coordinating with our operations group, fitting into its schedule and backlog.
- Replicated data in the mart would need to be maintained and controlled in addition to the original source data.
- Data security requires additional manual coding.
- Any changes required ETL scripts to be changed, as well as the mart to be reloaded, slowing our response to new requirements or even simple bug fixes.
- More data structure and syntax expertise was required by developers throughout the process, not just basic SQL.

SOA Data Services Approach Pilot Meets the Spec, Is Faster, and More

The data services approach proved ideal for our Drug Discovery Portfolio Portal project.

- We completed our project in less than half the time of traditional development. Much of the data-level development was automated, freeing our skilled development team to work on application-level development.
- Fewer skills were needed due to the drag-and-drop data service development environment, built-in security, and automated generation of Web data services.
- SOA-compliant WSDL data services provided data in the form the portal developers needed.
- Loosely coupled data services were easier to maintain than ETL scripts in case of changes either to the underlying data sources or the portal.
- Data service assets built for the portal project can be reused by other development projects.
- We no longer needed our IT operations team to build and maintain the data mart infrastructure. No extra costs for the mart itself.

Pfizer Informatics Adopts Data Services Approach

Going forward, we plan to use the data services approach and tools for all projects requiring complex data integration across multiple heterogeneous sources because the data services approach reduces unnecessary data replication and provides real-time information delivery, rapid application development, and SOA compliance.

We learned a number of lessons applicable to future projects. Data integration doesn't have to be hard or time-consuming with the right approach and right supporting tools. Virtualizing data versus replicating saves time and money. Rapid prototyping is possible, even automatic, when the right tools are used. Agility and reuse, the promise of SOA, comes to life in loosely coupled data services that span the gap between source data and end applications.

Moving from Pilot to Enterprise, Funded by Time and Cost Savings

With the new SOA data services approach to data integration proven, we have now put together our roadmap for future adoption. This roadmap includes educating our business analysts, developers, and architects on when to use data services and when to adopt the RAD approach to building SOA data services as the solution standard across all new SOA projects where data integration is required. Second, we plan to implement a "data services reuse" metric for measuring success across future projects to reduce development and maintenance costs. In addition, we're working with the centralized shared services team to create a Data Services Center of Excellence that promotes best practices, optimizes economies of scale, and maximizes reach across projects. Finally, we'll continue to seek emerging technologies and agile development practices that accelerate SOA projects and enable us to move to SOA in a safe and powerful way.

Conclusion

"Given the accelerating business demand for new systems from the R&D groups we support, my team decided to find a new approach to data integration that lets us build additional real-time solutions more rapidly, and in alignment with our SOA strategies, while avoiding the replication downside"

As advances in medical care and the need for new medicines continue to grow, the need for better ways to manage and deliver information is growing. In the same spirit that makes Pfizer a trusted leader in drug discovery and commercialization, the informatics group is pressing forward to meet the ever-demanding needs of our internal R&D customers as well.

Successful drug discovery needs data fast. To achieve rapid delivery requires new real-time portals and composite applications that rely heavily on existing data sourced from multiple systems from across the enterprise. Delivering that data to our researchers and managers has been one of our biggest bottlenecks, adding months and cost to our project timelines. These data integration needs, along with our aggressive SOA strategy and RAD objectives, have driven us to find, test, and deploy a new approach to data integration – SOA data services. ■

About the Author

Daniel Eng has over 17 years of diverse IT experience managing projects, leading technical teams, and developing enterprise applications at Fortune 100 companies. Currently at Pfizer Global Research and Development, Dan is leading efforts to transition business processes and applications to a SOA environment by using emerging technologies and agile management practices. Prior to Pfizer, he was an independent consultant helping his Fortune 500 clients develop intranet sites, portable applications, and e-commerce solutions. Dan has also worked in many e-commerce start-ups and healthcare organizations. He holds a BSEE from Polytechnic University and an MBA from Gonzaga University.

daniel.eng@pfizer.com

Sound Architecture Requires Proper Planning

WEB AGE SOLUTIONS - YOUR TRAINING PARTNER FOR SOA

In all phases of SOA migration, Web Age Solutions provides training and customized services from awareness to implementation. We support vendor specific or generic SOA training tailored to your organization's needs.

Custom training for complementary SOA technologies

XML • WEB SERVICES • WSDL • SOAP • WEBSPHERE • WEBLOGIC • JBOSS • J2EE • SPRING/HIBERNATE/STRUTS • WID/WBI/WMB

Custom training plans for virtually every job role

BUSINESS ANALYST • ADMINISTRATOR • DEVELOPER • ARCHITECT • QA/TESTER • MANAGER • EXECUTIVE • SECURITY

Consulting services for all phases of SOA migration

IBM Premier Business Partner

Add Web Age Solutions to your plan & stay ahead of the competition

www.webagesolutions.com - 877.517.6540 - soa@webagesolutions.com

WEB AGE solutions inc.

Legacy Systems

The risks & misconceptions

WRITTEN BY PAUL EVANS AND MARK LAJEUNESSE

One of a CIO's top priorities is to leverage information to enable better business decisions and outcomes. Many CIOs trace constraints on their business back to the limitations of older legacy information systems – specifically, outdated applications.

A legacy-based environment can make it difficult to attain the level of agility, flexibility, and responsiveness that growing enterprises need from their applications. For many organizations, the application has become the business – powered by technology. Modernizing applications is a key step to providing agility and lower cost of ownership, as well as creating a Service Oriented Architecture (SOA) and ultimately, accelerating business outcomes. However, modernizing also presents risks – both in the process and the final result.

To manage the transition safely from costly and inflexible application architectures to increased application agility – which is aligned to and able to respond to business demands flexibly – there are critical steps that enterprises must take. Through a modernization process, enterprises can build a flexible and manageable application environment that can take advantage of an SOA and shared IT infrastructure, to make the best use of their company's most valuable resources: people and information. Application modernization uses services and technologies to reduce computing costs and improve IT service delivery, while enhancing customer service and knowledge-sharing capabilities.

This article will describe drivers for modernizing applications, the common misconceptions, and critical steps businesses should take to safely modernize applications to create a path to SOA.

Why Modernize Applications?

- **Business Technology:** The new era of business technology represents a shift from information technology as a separate department to a model where technology powers the business. With it, the CIO is now also accountable for overall business outcomes such as managing risk, accelerating growth, or lowering costs. IT risks are now business risks, and IT opportunities are now business opportunities. By modernizing applications, the CIO can keep IT running smoothly, create new areas of innovation, and be an asset to the company.
- **Innovation and Cost Savings:** Today companies spend the majority of IT resources on application maintenance and support, leaving a much smaller portion for business innovation via IT. By modernizing applications, IT departments can achieve significant cost savings and devote that resource to innovation.
- **Desire for IT agility:** This desire is also driving organizations to modernize applications. An agile software infrastructure directly serves the needs of the business as well as the IT function, but it's proven that the business can have the most to gain. By reducing the lead time for application changes and by creating flexible, agile systems, the business can get the response from their IT systems that they've always demanded. Added to this is the cost savings that can be redirected from maintenance functions – up to 70% of IT costs in many organizations – to that of business innovation.
- **SOA is inevitable and will be a key architectural approach in the future:** Modernizing applications is a key step to creating an SOA. Companies today continue to invest in SOA to respond quickly to regulatory and competitive pressures. They also continue to adapt business processes to keep pace with market change. It's important to approach SOA with a combined software and services effort that provides a comprehensive foundation for successful adoption of SOA.

To achieve these outcomes effectively, CIOs need greater visibility into their application environment.

Modernization Fears

While most IT organizations are painfully aware that they have applications and systems that are underperforming, most aren't eager to undertake a modernization effort. There are several misconceptions surrounding modernization:

1. "This could compromise the service levels I'm accustomed to." Many times a senior IT executive will wonder, "If I move this piece of software from my legacy environment to a new system, will I get the same service levels?" Companies are hesitant to make the move and modernize their applications because they're unsure if the new system will garner them the same benefits.

The Right Road to the Best Candidates

Working with Two Roads Professional Resources means you benefit from our powerful MPC candidate matching system in these ways:

- Match top quality candidates with expertise in:

  Web Services
  Rich Internet Applications
  Application Development
  Gaming

- Partner with project leaders to understand what key roles are critical to the success of their projects
- Customize a comprehensive staffing solution for IT and Engineering

TWO ROADS
Professional Resources, Inc.

877-4-2ROADS
www.2roads.com

The key here is to understand which attributes of the legacy system are critical to the business need and ensure that the functionality and underpinning infrastructure deliver and exceed the quality of service that the business demands. The secret is not to 'blindly' move all of the legacy software to a new platform, but to assess what software is required and what can be replaced or discarded. It's worth pointing out that when the original legacy system was built, much of the functionality had to be written in code by hand. Whether it's using packaged software or operating system functionality, we can reduce the amount of handwritten code, thus providing a much lower risk to modernization.

2. "Modernizing applications is costly." Some CIOs aren't aware of the return on investment (ROI) that modernization can produce. Many IT leaders erroneously assume modernization requires a complete rebuild to achieve strong ROI, and very often that isn't the case.

3. "This will cause operational disruption and my business cannot afford that risk." CIOs fear that the shaky application "house of cards" they've built might crumble if it's revamped. They worry about potential security flaws, system downtime, and exposure to technical failure.

Part of any modernization effort is the transitional planning, which has to be based on a thorough assessment of the current state. The better we know the current environment and its linkages to other application components, the lower the risk in the transition. Once again, going in blindly is exactly the wrong strategy.

Steps to a Smooth Modernization

With the software environments customers have today, modernizing isn't a one-size-fits-all proposition. Unless the organization knows what must be done, and does it consciously, the change will have no meaningful effect. There are several key steps to consider when modernizing a company's legacy systems:

1. **Take inventory:** You've recognized that to achieve business goals, you need greater visibility into your application environment. The first step is to take an inventory of your applications – identify, analyze, and report on the status of the portfolio of applications in your environment.

2. **Prioritize:** To prioritize the application modernization project, CIOs should evaluate the existing architecture and constituent applications, review applications that may have differing characteristics, and examine historical and anecdotal information to determine the level of business value and technical quality of each application. Next, it's important to examine the potential costs and revenue impact of modernization for each application, and then rank and categorize each application as you discuss the best approach to take. Finally, identify those applications best poised for modernization and establish a target architecture.

3. **Create a detailed action plan:** Do a detailed analysis of the applications you've decided to modernize. This process focuses on specific applications and their underlying architecture – with a thorough examination of all the issues that affect three major tasks: 1.) existing application decomposability and cost impact on mining existing code; 2.) targeted application options including hardware, software, operating system, monitoring, management, and development; and 3.) incremental modernization strategy to move from the existing architecture to your target architecture, while allowing both to coexist during the transition.

4. **Transition your apps:** Once the roadmap is in place, the next step is to bring legacy applications up to today's performance standards while having a minimal impact on day-to-day business operations. These steps include re-engineering, re-hosting, replacing, retaining and retiring, as well as building, testing, and maintaining the new application environment.

5. **Finally, it's important to implement management and support best practices** to transition smoothly to the new operational environment.

Application Modernization Is a Must

Legacy systems stall innovation and are costly to maintain, but as any business professional knows, they run today's business. Modernizing applications isn't impossible if you plan properly. As a prerequisite for staying competitive in this global economy, IT modernization is a must-do for enterprises.

Companies are understandably concerned about the cost of modernizing its applications, the level of service during and after the move, and the possible exposure to technical failure. However, with proper planning and experience, it's possible to transition smoothly from legacy systems to an architecture that's more agile and responsive.

Enterprises should take an inventory of their application environment, and prioritize the most important and cost-effective applications. They can then analyze and determine the best course of action. Once a plan is in place, companies can begin the actual modernizing process through re-engineering, re-hosting, replacing, retaining, and retiring. Finally, a strong management and support system is key to a smooth transition. ■

About the Authors

Paul Evans runs the worldwide practice for application modernization for HP Services. Paul has worked in the IT industry for many years starting his career as a software developer before moving to positions with Digital, Compaq, and now HP. He has a passionate interest in future applications and their inherent architectures focusing on the business benefits that this technology can bring. This started with his work on the adoption of Ethernet-based networks through client/server to his current responsibility. Paul lives in the north of England.

Mark LaJeunesse is the Service Oriented Architecture (SOA) program manager for Hewlett-Packard Services' consulting and integration business. He is responsible for the development of SOA services, sales and delivery training, driving HP's EAS SOA visibility and differentiation in the market. Mark has 20 years in the high-tech industry. Prior to joining HP, he held a number of management positions at DEC, Compaq, Cybertech Systems, and most recently BEA Systems. At BEA he was responsible for running its SOA Program Office, where he and his team created BEA's SOA messaging, demand-generation programs and consulting services, as well as driving BEA's SOA visibility in the market through SOA industry forums and events.

OBJECT MANAGEMENT GROUP

ANNUAL SPONSOR:

Sparx Systems

SILVER SPONSORS:

Mega

Telelogic

Register before December 17, 2007 for early-bird savings and hotel discounts!

SAVE THE DATE!

OMG's Maximizing BPM Investments with SOA Workshop

January 14–17, 2008 • Orlando, FL

Maximize BPM investments while meeting real-world needs.

- Need to get your business folks talking to your IT folks?
- Need to know about how SOA can help you to maximize your BPM investments?
- Want to hear how other folks have succeeded?

This in-depth, three day workshop provides guidance and evidence on how business leaders and technologists can apply an SOA approach in developing IT support of their business needs and processes. Drawing upon OMG's experience in developing software standards, attendees will gain valuable insight through tutorials, keynotes, presentations and demonstrations on how industry leaders have worked through the hype and are developing BPM solutions that maximize investment while meeting real-world needs.

KEYNOTE JUST ANNOUNCED!

Maximizing BPM Investments with SOA

David Linthicum
Managing Partner
ZapThink LLC

Highlights include:

- In-depth tutorials detailing how SOA defines an architecture for business processes; modeling techniques for achieving business solutions; what standards mean to the business person; and the role of Event-driven Architecture (EDA).
- Informative sessions covering vertical market case studies and best practices; technical issues; strategic planning; and the "human factor."
- Expert panels exploring a model-driven approach to BPM and SOA and cooperation among standards organizations.
- A unique hands-on exercise—a team of pros working with the audience to solve a sample business problem using available technology.
- State of the art technology exhibits and cutting edge demonstrations from a wide variety of solution providers.

Presenters will represent leading organizations such as:

Adobe Systems	IBM	PricewaterhouseCooper
BEA Systems	IDS Scheer Global	Relativity Technologies
Burton Group	Interfacing Technologies	SOA Consortium
Collaborative Consulting	Kohl's Department Stores	The RuleML Initiative
CSC	Lombardi Software	TIBCO
Deutsche Bank	ModelDriven.org	VISA
Elemental Links	OASIS	W3C
Harvard Computing Group	OCTL Consulting	*and many more!*

For more information or to register:
ONLINE: www.omg.org/bpm-soa-wm
CALL: +1-781-444 0404
EMAIL: loughry@omg.org

Don't miss this unique event! Register before December 17, 2007 for early-bird savings and hotel discounts. Visit www.omg.org/bpm-soa-wm for complete details.

Where Are Your Business Rules Anyway?

Forming business rules into standalone business services or Decision Services

WRITTEN BY DAVID STRAUS

The primary purpose of most business applications is to bring automation and efficiency to the business. Outside of a few examples, most of our business applications are required to evolve along with our business, and that, of course can be a challenge for most organizations

The time it takes to change application software gates our business's ability to remain competitive or stay current with internal or external initiatives or regulations. This article is about the core asset in our applications that deals with our business: our business logic or business rules. We will discuss how to group these business rules together to form automated business decisions that can be easily created and maintained.

So where are your business rules? In many instances, most of our business rules exist in one of two places: in the heads of our employees (hopefully in the context of formal policy or procedure manuals) or in our applications, our legacy applications. When I asked Forrester analyst Phil Murphy to define what constitutes a legacy application, he said, "Anything that was written more that 18 months ago." Once programmed, applications become "legacy" fairly quickly. When these business rules are buried in "legacy" applications, problems exist for the both the business organization and IT. Problem one is agility, or the ability to change the application to meet the current business need. Problem two is transparency, or the ability to know exactly what the application is doing in a reliable and auditable way. A recent Forrester report indicated that more than 75% of the IT budget is allocated to application maintenance leaving little for new application development. These problems create friction in many organizations[1].

Along comes SOA or Service Oriented Architecture. Explaining SOA is an article onto itself but in a nutshell in a SOA-based application the business rules associated with key business decisions (like rules that determine claim approval) are formed into standalone business services or, in our case, Decision Services. We will discuss the lifecycle of finding and transforming these critical business rules from your legacy applications to well-formed Decision Services. Decision Services let you maintain the logic in a more agile and transparent way and gain the benefit of greater reuse. These new Decision Services can be called by your legacy application or can be used in new SOA-based application frameworks (e.g., Business Process Management Systems). We'll summarize three topics focused on getting to this tremendous base of legacy business logic:

- Knowledge mining of legacy applications to surface your business rules
- The value of transforming business rules into business Decision Services following the precepts of a Service Oriented Architecture
- Methods for transforming your business rules to modern agile business Decision Services

Knowledge Mining

Most organizations I've talked are concerned about what's in their legacy code. In fact, when we have used knowledge mining techniques to determine exactly what's in their code, most are surprised. Knowledge mining is the process by which legacy code is analyzed and documented. Knowledge mining can be accomplished manually (people reviewing code), but it can be done more effectively by using software tools to parse and map every element of an application. I'll list a set of some of the key outputs you should expect from a knowledge mining exercise.

- You should get a clear map of program flow and control including application overview maps, flowcharts, and UML diagrams,
- Get a map of all variables in the code that identifies which are business variables (relating to business rules) and which are control variables (relating to internal control structures),
- Get a listing of your business rules, including the organization of the business rules into logical functions and sub-functions.

Note on Business Process

I'm often asked if knowledge mining solutions can surface an organization's business processes. For most solutions I've seen I would have to say no. Business process wasn't an explicit design in most legacy applications. User interfaces exist and, for the most part, employees know which screen to use when. I've seen this at banks, for example, where the employee moves from one application to the next to go through the process of opening a new account for a customer. The process can be different depending on whether or not an employee is dealing with a new customer. Legacy applications are rarely explicit about the business process. There are some interesting new capabilities emerging that mine data and use inductive principals to form the business process based on changes in data over time (following the business process lifecycle of the data). One such tool is available from the business process management company IDS Scheer.

Transforming Business Rules into Decision Services

We've seen organizations knowledge-mine their business rules only to transform them from one language (e.g. COBOL) to a different language (say, Java). The problem with this strategy is that you've created your next legacy application problem, almost before you've completed your work. There are two concepts that you

should entertain in considering what to do with your legacy business rules. The first is using a business rules management system (BRMS). The second is forming them into discrete business services or Decision Services

A business rules management system is a solution that helps form your business rules into independent Decision Services and then manage the lifecycle of change and promotion of these Decision Services into production. Most notably, a BRMS will enable rules creation and maintenance in a way that allows the business organization (think "Business Analyst") to create and modify its own business rules. To accomplish this sharing of IT assets a BRMS will have a user interface that's "business-friendly," will have a rules management repository that controls access to business rule change, and will manage the approval and promotion of the changed Decision Service into production.

The value of SOA-based services is described above, but again includes agility, transparency, and reuse. Following the principals of a SOA, each Decision Service is atomic and can be changed without impacting its service contract in most cases. This enables significant agility as the core logic can change, but the application(s) in which it's used remain stable and unchanged.

Methods for Transforming Legacy Applications Logic to Agile Decision Services

Most knowledge mining tools or solutions on the market produce reports (knowledge), often based on a database of information mined from the legacy application. These reports are quite useful and go a long way toward helping you understand your application. For some organizations these reports become a critical tool to help in the software maintenance process. For others they become the basis or starting point for transforming their business rules into well-formed Decision Services. The best solutions help to automate this critical transformation. There are companies that form their solutions around these tools and others that use a more manual process combined with low-cost offshore resources to create a similar cost to knowledge mining and business rules transformation.

Summary

The number of legacy applications has never been estimated, but based on many discussions, the pain is clear. Knowledge mining business rules and transforming them into effective Decision Services provide a clear path to SOA and to significant business value. But this is not a simple process, so make sure your solution provider has the experience and proven results and doesn't just move you from your existing legacy application to your next one.

Resources

1. Phil Murphy. "CIOs: Attack Weak Application Maintenance Processes That Stifle IT Productivity." Forrester Group. February 12, 2007 (http://www.forrester.com/rb/analyst/phil_murphy)

About the Author

David Straus is responsible for Corticon's global marketing, which includes product management and marketing, field marketing, and corporate communications. he joined Corticon with over 20 years of enterprise software solutions experience in product marketing and sales. He has held executive positions at Chordiant Software, which acquired OnDemand, Inc., a company he founded in 1997. David graduated from Indiana University with a BS in business and operations research.

Legacy Transformation and SOA

Modernization, the no brainer

WRITTEN BY **JOHN BILLMAN**

Despite the IT industry's obsession with the "next big thing," even valuable innovations are often put into context by legacy systems. Your bank account, mortgage, home and auto insurance, and pension are all likely to be running on a so-called legacy system.

They typically run a large portion of a company's critical business processes and contain years' worth of valuable coding and enhancements. While most of these systems are clearly getting the job done, they must evolve to continue to support the business in the most efficient manner.

As pressure mounts to access core functionality from the Web, integrate heterogeneous systems across the enterprise, comply with regulatory requirements or support new lines of business, legacy systems can no longer operate in isolated silos.

So, how can IT executives preserve mission-critical functionality while keeping pace with the latest and greatest in IT? There are three options that they typically consider:

- **Rewrite** — build a new system in a contemporary programming language to meet business requirements
- **Package** — purchase a new package that implements traditional core functionality
- **Modernization** — extend existing legacy systems to meet new requirements

There's no single correct answer as to which choice is correct. It can depend on the nature of the legacy system and the emergent business requirements – and in some cases, a combination of approaches may even be appropriate.

On the whole for a non-trivial enterprise system, the "rewrite" option would almost always represent the greatest risk and highest-cost approach. A rewrite is quite a different proposition to a "new write" of a completely fresh system that didn't previously exist. "Rip and Replace" involves spending considerable effort simply to replicate the functionality already available today. This approach requires a difficult combination of existing application and new technology knowledge. As we are all painfully aware, most large-scale projects of this kind will face challenges with timescales, cost, functionality, or performance.

A package, in contrast, will contain a core of proven functionality. However, a careful "gap analysis" of the package functionality and existing system functionality often reveals a need for significant extension in addition to a major new cost.

So, while both rewrites and packages have their place, in most scenarios the modernization and extension of an existing legacy system to meet new requirements will represent the fastest, most cost-effective and lowest-risk approach. The existing application can remain unchanged, meaning current business operations are not affected at all. As a result, the scope of the project can remain tightly defined to meet the new requirement at hand. Modernization projects carry lower costs than a rewrite because they reuse critical business logic proven and evolved over many years. In addition, modernization makes use of the existing skills of mainframe programmers when pursuing new business requirements. It's important to draw on the expertise of these seasoned developers — much of which would be lost if legacy applications were disregarded.

Enter SOA

There's a dichotomy present in IT shops across the globe: preserve worker knowledge and application operations but embrace innovation, new technology, languages and delivery platforms. That's why one of a CIO's greatest priorities is to help his or her organization increase its market share and stay competitive by using the newer technologies – without sacrificing the benefits delivered by reliable legacy applications. In light of this, using Service Oriented Architecture (SOA) as a path to modernization is a major consideration for IT executives.

They take notice when they learn that, by migrating to a SOA through a Web Services environment, they can cut costs, reduce risk, and hasten time-to-market for new business initiatives. As such, making the move to SOA through reusable Web Services encapsulating legacy business processes represents a standards-based, strategic, and flexible way to modernize legacy applications.

The value and benefits of SOA are well publicized today but often in the context of new development. However, when applied to legacy systems SOA can become truly compelling. SOA frees a business process from the language in which it was written, as well as the platform it runs on – a powerful value proposition. This means that organizations can combine the value of proven business functionality with the agility of reusing services and the capabilities of .NET and other contemporary programming environments. Advantages of legacy transformation through SOA include:

- Improved agility through the reuse of existing business processes — any future compliance mandates or business needs can be addressed by reusing services previously updated, regardless of the platform or delivery
- Low-risk operations through consistency and reliance on proven business logic
- Reduced costs as middleware, software, and integration costs are lowered through the use of a single strategic architecture
- Improved time-to-market through rapid development enabled by reusing existing business processes in a standard SOA infrastructure

Put into Practice

Of course, with 200 billion lines of COBOL in production and legacy systems typically comprising tens of millions in lines of code, a complete infrastructure overhaul via SOA and Web Services would neither be practical or sensible. Most successful SOA initiatives are incremental, focusing on the business need and encapsulating the processes that present that business functionality as services. This approach can also force the discipline of actually defining, in isolation from the legacy system, the services that you'd like to create. In other words, the business need should define the strategic service framework rather than the legacy system simply replicating its existing interfaces as a potentially complex array of services.

Once the required services are defined, the areas in the legacy system that deliver this functionality must be identified. At this stage, a combination of application development knowledge and the use of program-understanding tools or solutions is highly recommended.

What follows is the heart of this whole process – the "unlocking" of legacy applications by leveraging their existing interfaces. Mainframe COBOL, CICS, and IMS systems have a variety of interfaces such as 3270 Data Flows, CICS COMMAREAs, and Screens that can be utilized. These are the way into the legacy code.

This is the stage where it becomes important to understand the new business need and the service definitions that best service this need. Rather than simply replicate the legacy interfaces, they should be aggregated and tailored to combine multiple transactions from the legacy system into the new business service.

Once this definition process is complete, this service can be deployed as an industry standard Web Service into .NET or a J2EE application server. Legacy functionality can now be accessed from a contemporary .NET user interface created and maintained using Visual Studio. Simply consumed as Web Services core business processes can be accessed from .NET languages, and integrated with new development under .NET or any other technology capable of consuming Web Services.

Benefits Unleashed

Beyond the obvious benefits of modernization, there are several underlying advantages to transforming legacy applications through SOA. By enabling enterprise systems to address new requirements or comply with new initiatives, and ensuring that these extensions don't alter or affect existing systems, valued functionality and intellectual property are retained. This precipitates a positive domino effect. It frees up time for businesses to focus efforts and resources on new projects, optimizes time-to-market for new products, and at the same time controls costs and reduces risk.

Another benefit of extending legacy systems to a SOA through Web Services is the ability for strategic business assets to be accessed at any time from any location. This makes cross-enterprise (or even extra-enterprise) integration a seamless process.

While IT decision makers should carefully evaluate the commitment to modernization through SOA, there are many positives to the approach. It is clear that SOA's strength lies in providing a standards-based infrastructure and a platform neutral way of delivering business services. The ability to retain the valuable functionality contained within legacy mainframe applications while at the same time defining and deploying services from them makes modernization virtually a "no brainer."

Ultimately, an SOA approach to application modernization is a low-risk, cost-effective path to ensuring that key enterprise systems evolve to best serve the business.. ■

About the Author

As a Product Director at Micro Focus John Billman has special responsibility for Application Modernization and is responsible for both evangelizing the technology while setting direction and market requirements for future versions. He has extensive experience of Product Management and Development and joined Micro Focus in 1988. At Micro Focus John has worked on multiple projects including the creation and launch of the first Micro Focus product for .NET. Prior to Micro Focus he was a mainframe programmer and gained a degree in Computer Studies.

SOA's Last Mile

Enterprise Web 2.0 bridges the gaps in SOA for greater business agility

WRITTEN BY **CLIFF LEE AND KEES NEVEN**

Combining Enterprise Web 2.0's (EW2.0) ability to deliver business-critical applications over the Web and Service Oriented Architecture's (SOA) capability to deliver services provides a solution that offers the ubiquitous consumption of service for anyone anywhere in any environment.

What to do with this ability will be the next challenge for IT and end users. IT will look for ways to create service-based components that can be reused across the enterprise. End users will look to be able to create their own "mash-ups" service components that will best suit their needs. Enterprises will continue looking for the return on their investment in their SOAs.

The key to meeting everyone's needs is a solution that lets organizations leverage legacy, Web Services, and SOA-oriented data sources to build enterprise-class "mash-up" or composite applications that extend beyond the data located behind firewalls or provided through a service vendor to include publicly available data from the Web. This means IT managers can quickly and easily add Web-based data, in turn giving users more data to work with, helping them do their jobs better and faster, and ultimately supporting businesses be more profitable and successful.

SOA

Companies today recognize the need to align IT assets quickly to meet changing needs. To stay competitive, organizations have to be agile; the data and the applications that support businesses must have the flexibility to change on-demand. Application administrators or end users must be able to customize the application or create a composite application to support specific situations.

SOA is a design and business strategy that facilitates this change by providing a highly adaptive software system that supports new applications that are loosely coupled, flexible, and highly agile.

Using SOA, IT organizations can create applications and access information using standards-based, reusable business services that can be directly mapped to business functions. Once an enterprise has exposed its services, it may then design new composite services or business applications and processes to align further with the organization's mission.

Fundamental concepts of SOA include:

- Everything is considered a service: business service (e.g., CRM, order processing), infrastructure service (e.g., security, transaction management), and UI service (e.g., mash-ups, widgets).
- All services are directly addressable by applications or other services.
- Services are interoperable using standards-based loose coupling.
- Applications that consume services are insulated from changes in other applications.

The successful implementation of SOA will increase an organization's responsiveness to changing market demands. It mitigates the risks associated with vendor lock-in and reduces development costs by leveraging legacy assets and reusing existing services. It helps drive new revenue and increases market share as well as time-to-market.

For many companies, their SOA initiatives have reached an inflection point. Most best-in-class companies have found the SOA infrastructures in place have successfully unlocked valuable business information from disparate back-ends.

The challenge that remains is to deliver these business services to the end user. This "last mile" of SOA needs to be delivered for the IT department to fully reap the benefits of its efforts. Enterprise mash-up applications are emerging as the preferred solution to empower the end user with SOA-enabled applications. In this model, organizations focus on the consumption layer of SOA and its extension into the client tier.

This could also mean that the UI widgets themselves are services. These pre-built widgets will already be connected to data and services; and they will be ready for consumption.

Enterprise Web 2.0

Web 2.0 promises to turn the Internet into a true operating platform — featuring robust client-side logic and rich interfaces that mimic the performance and security of fat client applications. For enterprise IT teams, achieving the aims of Web 2.0 requires more than adopting popular Rich Internet Applications (RIAs) development languages like AJAX, Flash, Java, and .NET.

Enterprises require:

- A framework that provides standardization and simplification across different business applications and development technologies while ensuring the flexibility required for innovation within business units. It must also support SOA initiatives by enabling the consumption of loosely coupled services that provide access to business functionality and data in real-time
- Applications with reliable and secure communications between client and server operating across any network, browser, and operating system and must include a distributed model-view-controller that reduces server calls and performs consistently whether online, offline, or mobile.
- A development environment that leverages existing code, development standards, tools, skills, and infrastructure.

EW2.0 requires an architecture that integrates both an Enterprise Services Layer and a Services Consumption Layer.

SOA and Enterprise Web 2.0

Many characteristics of the Enterprise Service Layer also apply to the consumption layer. UI services and widgets are pre-built and highly reusable, similar to back-end application services exposed as business services. Preferably, they are bundled together to form a composite application where the widgets are already connected to the back-end business services. These "service widgets" form the ideal components to create mash-up applications.

These widgets are governed like business services in an SOA. Providing the visibility, lifecycle and change management of these widgets is important to ensure quality, predictability, and transparency.

Existing SOA governance platforms (SOA Registry and/or Repository) focus on these key areas as they relate to server-side business services. The best platforms open the dialog with different SOA vendors to achieve interoperability.

To achieve the benefits of an SOA-enabled EW 2.0 application, the services should be governed and easily discoverable for architects, business analysts, developers, and users. By doing so, development of applications will be more consistent across the enterprise. Developers will be able to browse through the list of services from their integrated development environment (IDE) and consume both business services and UI services.

This governance will also allow service providers to perform impact analysis. By analyzing the relationships between the consumer and the provider of services, the services can be continuously improved. Examples of such relationships include service widgets consuming business services, "mash-ups" consuming service widgets or mash-ups, and composite services consuming business services. Conditions such as security compliance policies, standards, and service level agreements (SLAs) must be defined for each relationship. The quality and performance of the Web application depends on the quality and performance of the underlying services.

SOA governance platforms will manage the relationships and the associated policies and service levels. As a result, impact, lifecycle, change, quality, and performance management processes will apply to both the UI and business services. ■

About the Authors

Cliff Lee is the director of technical services for Nexaweb Technologies, Inc., where he is responsible for pre-sales and consulting services. Prior to joining Nexaweb, Cliff was responsible for developing the professional services organization at MarketSoft, enabling the successful delivery of lead and campaign management solutions for Fortune 500 companies such as Fidelity, Bank of America, American Express, and Mass Mutual. Before MarketSoft, Cliff held consulting positions with Accenture where he delivered solutions using a variety of technologies in multiple vertical markets.
clee@nexaweb.com.

Kees Neven is the managing director of Northern Europe for Nexaweb Technologies, Inc., where he is responsible for territory sales and consulting services. Prior to joining Nexaweb, Kees catapulted Systinet into Northern Europe by building and running the pre- and post-sales operation. Prior to Systinet, Kees held positions with industry leaders such as webMethods and Cisco. He has been referenced frequently in leading Dutch IT magazines such as Computable.
kneven@nexaweb.com.

About Nexaweb

Nexaweb's Enterprise Web 2.0 Suite is an application development and deployment platform for building mission-critical, secure Web-based business applications. With Nexaweb, building enterprise-class Web applications that access legacy and service-oriented data systems is simple and doesn't mean rewriting code. And only Nexaweb's EW2.0 solution is proven by more than 5,000 successful global deployments. Founded in February 2000 and rooted in the MIT community, Nexaweb is based in Burlington, Massachusetts. The company is privately held and funded by institutional investors. For more information, see www.nexaweb.com or call (781) 345-5500.

SOA and Mass Data Movement via ETL

– CONTINUED FROM PAGE 6

Here is a step-by-step definition of the sequencing of calls as shown in Figure 1 that represents how SOA and ETL can work together in the real world. Figure 1 also shows how the architectural components are wired together to achieve the goals of the pattern:

1. The Service Consumer calls the Enterprise Service to submit a request (scheduled) or invoke a service in real-time
2. The Enterprise Service checks to see if this information exists in its Enterprise Staging Repository
3. If the information isn't available, standardized ETL processes are executed (on-demand/scheduled) that gather the relevant legacy data, transform, and load the complete enterprise-worthy business entity information into the Enterprise Staging Repository. The ETL process also applies enterprise-worthy transformation and data reconciliation rules to the legacy data prior to uploading the Enterprise Staging Repository.
4. The Enterprise Service executes enterprise-worthy business rules and processing logic to perform a commonly used business function on behalf of the consumer
5. The Service Consumer is notified of the result of the execution of the business function

This pattern was applied by me at a large retailer to deal with "collecting" and "populating" merchandise assortment information from various vendor merchandise assortment repositories and private brand merchandise assortment repositories. The ETL process transformed the relevant information regarding various types of merchandise assortments prior to uploading the enterprise repository. This enabled the service provider layer to apply enterprise-level "merchandise assortment rules" to satisfy the "optimize merchandise assortment by region" requests.

In conclusion, the attempt made in the architectural pattern is to show how an enterprise can extend its legacy information assets while insulating the consumer layer from the details of the process. The legacy systems continue applying localized rules to capturing the data while the enterprise service only adds on the enterprise rules layer thus avoiding redundant application of business rules. In addition, front-ending the legacy data sources with a SOA-style service lets the SOA service management infrastructure be leveraged to manage and monitor consumer SLAs without affecting the fragile and customized legacy application code base. Thus, an enterprise that has legacy assets can still move ahead and apply the core principles of SOA such as loose coupling and modular design without having to sacrifice the stability of the legacy systems or decommissioning them.

The result is SOA-style services that can now expose enterprise information to satisfy any business process whether it's from within an enterprise context or from an extended enterprise business context.

Finally, the key value provided by this pattern is to demonstrate how enterprises can embark on the SOA path feeling empowered by the portfolio of legacy assets they have at their disposal instead of looking at them as a liability. ■

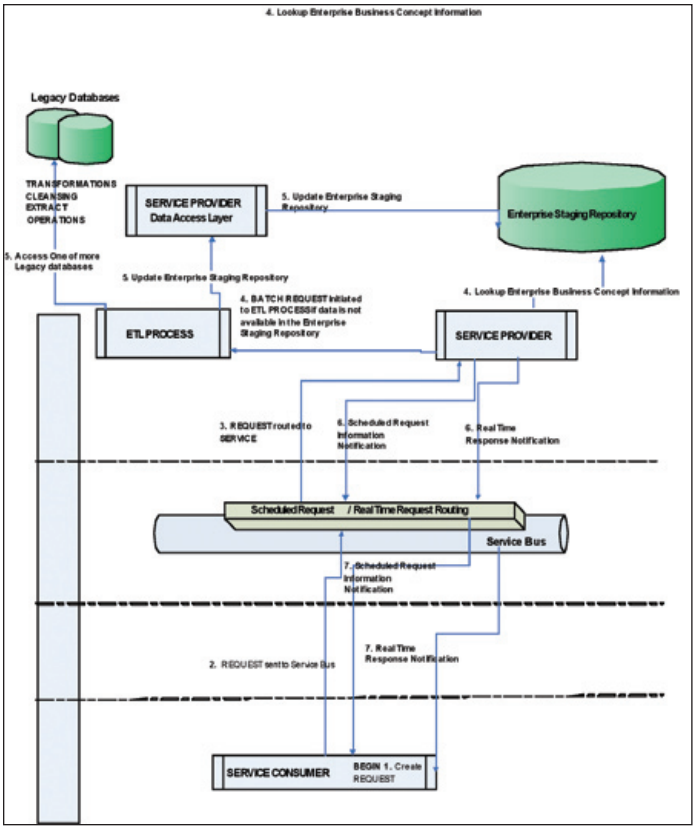


Figure 1:

Advertiser Index

ADVERTISER	URL	PHONE	PG
Active Endpoints	active-endpoints.com/soa		36
Altova	www.altova.com		19&35
Attachmate	www.attachmate.com/soa		11
IBM	IBM.com/takebackcontrol/soa		2&3
Managed Methods	www.managedmethods.com		7
Mellanox	www.mellanox.com	408-970-3400	15
Mindreef	www.mindreef.com	603-465-2204 ext.581	5
Nastel	www.nastel.com		13
Object Management Group	www.omg.org/bpm-soa-wm	781-444-0404	25
Production Hub	www.productionhub.com	877-629-4122	21
S D I, Inc.	www.shanghai-digital.com		27
SYS-CON Books	books.sys-con.com	201-802-3020	33
Two Roads	www.2roads.com	877-4-2ROADS	23

SOA Vendors Need Some Coaching

Taking the right approach

WRITTEN BY **DAVID S. LINTHICUM**

Last month I wrote about vendor-driven architectures (VDA), and I had a few vendors ask me to look on the other side of the fence. In essence, to consider how vendors can better address the needs of the customer, considering the new drivers with SOA.

Truth be told, I can't believe the unsophisticated approaches many vendors have when selling their product. Indeed, I'm taken aback weekly by a vendor pitch that just does not flatter their technology, perhaps even making them take a few steps back in my book, and perhaps in the opinions of their customers.

Core to this is the fact that many SOA vendors can't explain their own product, or the core problems it solves. They do know how to list buzzwords they think will "wow" their prospects and existing customers. However, in many cases, the customers become further confused or, worse, don't even get the core concept behind the product, not to mention SOA.

Case in point, many vendors, when asked about their closest competitor's products, have a very well-rehearsed response, and point out (spinning really) the differences between their offerings... In essence, they explain how the other guys "are really bad" and we "are really good." Meanwhile, in another conference room far away, the same conversation is occurring, but in reverse.

Unfortunately, the sales teams, even those armed with the smartest SEs, fail to deliver more than a very canned and ineffective pitch and/or briefing, and end up looking bad and confusing people they should really not confuse. This is not a trend; it's an outright epidemic.

What do you do? The right approach to this problem is something that many vendors don't even think about until it's too late. The core pitch should be around how the product solves a customer's specific problems, as well as a detailed, easy-to-understand approach to the "solution." Even (gasp!) tell them what problems you don't solve, and perhaps recommend other products that provide a better fit.

You start, however, with an understanding of the customer issues, including a quick and dirty intro into SOA at a holistic level but narrowed eventually to their vertical. Then, drill down into their problem domain (a.k.a., project), and then and only then identify pain points that your product could resolve, and how, specifically, you can do that...Step 1, 2, 3, etc.

At the end of the day, it's just a matter of matching problem patterns with solution patterns, thus looking at what the core issues are that the SOA needs to address, and then determining which technology is right for those patterns. While many believe that there are SOA-in-a-box solutions, there really is no such thing, and thus the architecture world in SOA needs to take precedence. Indeed, requirements that I see around SOA are all very different, and thus so should be the technology solutions. While it would be a nice world if a single vendor would always be the right fit, those are actually few and far between.

SOA vendors need to embrace a more consultative type of selling approach. So, the vendors that will succeed will have the heart of a teacher, not a salesman. They need to arm those who are going to sell the technology with a clear understanding of the attributes of SOA, and learn how to listen to the core issues around the business, as well as learn how to drill down on the real issues that the customer may not be telling you. For instance, I often hear how well their architecture is currently working, but, upon further analysis, find that there are major flaws that need to be addressed, typically around the agility of the current architecture, or the ability to adjust to changing business requirements.

The main point is that this is all very new. Most vendors have never sold an architecture before, just tactical products that service some specific purpose. All architectures, inclusive of SOA, are really around the right configuration of technology and understanding, and not technology itself. That's a huge change for many, and I suspect most will fail when attempting to change their approach. Now is the time to get some help, figure out how you go-to-market, and learn to love your customers long term. ■

About the Author

David S. Linthicum (Dave) is a managing partner with Zapthink, LLC (www.zapthink.com), a consulting and advisory organization dedicated to excellence in SOA planning, implementation, training, mentoring, and strategy. He 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.
david@linthicumgroup.com

Altova® XMLSpy® 2008 – The industry standard XML development environment.

ALTOVA® xmlspy® 2008

Immerse yourself in XML intelligence

Dive into XMLSpy®, and take XML development to greater depths.

Spied in Version 2008:

- Support for Microsoft® Word, Excel, and now PowerPoint Open XML formats
- Direct editing of database content via database query window
- XPath evaluation across multiple files at once
- Support for XInclude and XPointer

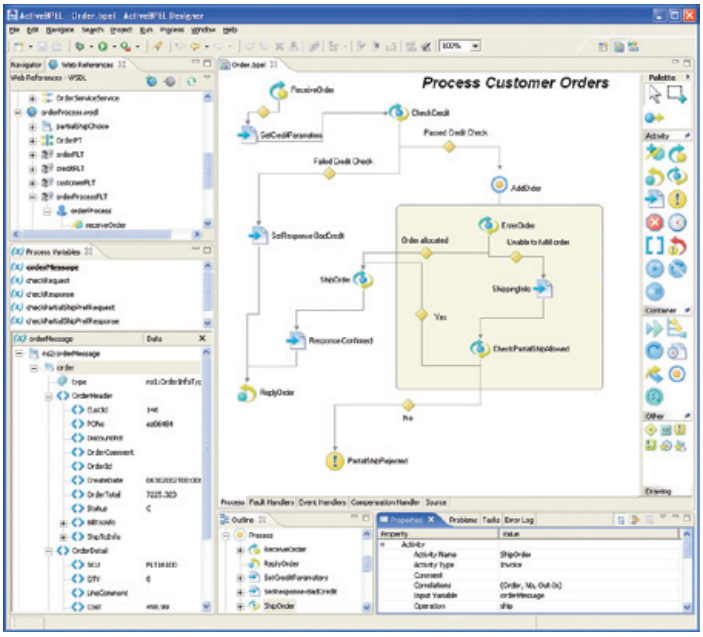
Altova® XMLSpy, the industry standard XML development environment, is vital for modeling, editing, transforming, and debugging XML applications. Breathe life into your plans with the world's leading XML editor, the original graphical schema designer, a code generator, file converters, debuggers, profilers, full database integration, support for XSLT, XQuery, WSDL, SOAP, and a trove of treasured XML utilities and usability aides. Become a markup mastermind!

Download XMLSpy® 2008 today:
www.altova.com

XMLSpy is also available as part of the value-packed Altova MissionKit™ software bundle.

ALTOVA®

Get Started with BPEL 2.0



Build next-generation SOA applications with the leader in BPEL technologies

Download BPEL tooling & server software today

active-endpoints.com/soa

BPEL consulting and training.
BPEL design tools, servers and source code for Eclipse, Apache Tomcat, JBoss, WebSphere, WebLogic, BizTalk and Microsoft .NET.

activeBPEL

active endpoints