

WebSphere® JOURNAL

www.WebSphere.SYS-CON.com

The World's Leading Independent WebSphere Resource

MAY 2006 VOLUME 5 ISSUE 5

Where Has My Data Gone?

*Accessing Data in a Service
Oriented Architecture*

BY CLEMENS UTSCHIG
AND DOUG CLARKE
PAGE 6



The Challenges of SOA

BY DAN FOODY
PAGE 2

The Challenges of SOA

Which rules are necessary and which are just nice to have

BY DAN FOODY

“Our processes are bulletproof. Nothing gets into production that doesn't go through the proper and complete approval process.” Famous last words uttered by far too many enterprise architects. Some of them actually believe it's true – others think that by hoping it's true, maybe, just maybe, they can make it true. The reality, as any line-of-business developer can attest, is much less clear-cut. The challenge is that governance only gets harder the more an organization moves towards a service-based architecture.

One of the first myths that drives a number of enterprise architecture governance decisions is that adding more rules reduces risk. That may be true in theory, but in practice it actually increases risk. The reason is simple: complexity increases risk. A perfect case study of this, one that most people have probably experienced, is password-control policies. As many IT organizations have attempted to “improve security,” they've done things like disallow use of dictionary words in passwords, force passwords to change often, disallow reuse of older passwords, etc. The net result is that, because of the added complexity, more people write down their password on a Post-it note. And written-down passwords increase

“A myth that drives a number of enterprise architecture governance decisions is that adding more rules reduces risk; that may be true in theory, but in practice it actually increases risk.”

the likelihood of a security breach while, at the same time, making it harder to detect the breach. Increased complexity increases risk.

Avoiding the Complexity Pitfall

There are two ways to address this complexity issue:

- Have fewer rules, but make them more important rules
- Automate compliance with the rules

In terms of gauging the importance of rules, I've seen a number of cases where architects put too much emphasis on the technical side and too little emphasis on the business side. For example, let's look at a technical requirement: the need to promote reuse. This often leads to many rules: Rules around the use of certain schemas, security mechanisms, designing a service interface, and many others. Reuse is no doubt important so it makes sense to have rules to promote it. But, let's contrast this with a business requirement: regulatory compliance – whether it's Sarbanes-Oxley (SOX), European Union privacy regulations, HIPAA, or even Visa's Cardholder Information Security Program (CISP). These lead to a large set of rules as well. So, let's say you had to choose between rules to promote reuse or rules to ensure regulator compliance. Would you choose the rules that have no directly quantifiable upside and, at worst, lead to increased cost and reduced agility? Or, would you choose the rules that would keep you from going to jail, getting fired, getting fined, or force your company to shut down? When put in these terms it's easy to see which rules are the most important.

The other approach is to attempt to automate as much rule checking as possible. There are solutions that help address this at every stage of the application lifecycle. Of course, not every rule can be automated, so you still need to be mindful to tightly control and prioritize the set of rules that development must follow.

ADVISORY BOARD

Richard Arone, David Caddis, Mike Curwen, Devi Gupta, Lloyd Hagemo, Barbara Johnson, Shannon Lynd, James Martin, Doug Stephen, Victor Valle

EDITOR-IN-CHIEF ROGER STRUKHOFF
CONTRIBUTING EDITOR PATTI MARTIN
PRODUCT REVIEW EDITOR JAY JOHNSON
EDITOR NANCY VALENTINE

SUBSCRIPTIONS

For subscriptions and requests for bulk orders, please send your letters to Subscription Department.
888-303-5282 201-802-3012
SUBSCRIBE@SYS-CON.COM
Cover Price: \$8.99/Issue Domestic: \$149/YR (12 Issues)
Canada/Mexico: \$169/YR Overseas: \$179/YR
(U.S. Banks or Money Orders)
Back issues: \$15 U.S. \$20 All others

PRESIDENT AND CEO	FUAT KIRCAALI
PRESIDENT, SYS-CON EVENTS	GRISHA DAVIDA
GROUP PUBLISHER	JEREMY GEELAN
SENIOR VP, SALES & MARKETING	CARMEN GONZALEZ
VP, INFORMATION SYSTEMS	ROBERT DIAMOND
VP, SALES & MARKETING	MILES SILVERMAN
FINANCIAL ANALYST	JOAN LAROSE
CREDITS & ACCOUNTS RECEIVABLE	GAIL NAPLES
ACCOUNTS PAYABLE	BETTY WHITE
ADVERTISING DIRECTOR	ROBYN FORMA
ADVERTISING SALES MANAGER	MEGAN MUSSA
ASSOCIATE SALES MANAGERS	LAUREN ORSI
	KERRY MEALIA
LEAD DESIGNER	ABRAHAM ADDO
ART DIRECTOR	ALEX BOTERO
ASSOCIATE ART DIRECTORS	ANDREA BODEN
	TAMI BEATTY
	LOUIS F. CUFFARI
VIDEO PRODUCTION	RYAN PALMIERI
WEB DESIGNERS	STEPHEN KILMURRAY
ONLINE EDITOR	ROGER STRUKHOFF
CIRCULATION SERVICE	
COORDINATOR	EDNA EARLE RUSSELL
CUSTOMER RELATIONS MANAGER	BRUNI STAROPOLI

EDITORIAL OFFICES

SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
WebSphere® Journal (ISSN# 1535-8914)
is published monthly (12 times a year).
Postmaster send address changes to:
WebSphere Journal, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
CURTIS CIRCULATION COMPANY, NEW MILFORD, NJ

NEWSSTAND DISTRIBUTION CONSULTANT:
GREGORY ASSOCIATES, W.R.D.S.
732 607-9941, BGJASSOCIATES@CS.COM

FOR LIST RENTAL INFORMATION:
KEVIN COLLOPY: 845 731-2694, KEVIN.COLLOPY@EDITHOMAN.COM
FRANK CIPOLLA: 845 731-3832, FRANK.CIPOLLA@EPOSTDIRECT.COM
© COPYRIGHT 2006 BY SYS-CON PUBLICATIONS, INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM, WITHOUT WRITTEN PERMISSION. FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR, CARRIES@SYS-CON.COM. SYS-CON PUBLICATIONS, INC. RESERVES THE RIGHT TO REVISE, REPUBLISH AND AUTHORIZE THE READERS TO USE THE ARTICLES SUBMITTED FOR PUBLICATION.

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES. SYS-CON PUBLICATIONS, INC. IS NOT AFFILIATED WITH THE COMPANIES OR PRODUCTS COVERED IN WEBSphere JOURNAL.

WEBSphere® IS A REGISTERED TRADEMARK OF IBM CORP. AND IS USED BY SYS-CON MEDIA UNDER LICENSE. WEBSphere® JOURNAL IS PUBLISHED INDEPENDENTLY OF IBM CORP., WHICH IS NOT RESPONSIBLE IN ANY WAY FOR ITS CONTENT.



Fireproof Your Code

Prevent Java code fires with JProbe®

Constantly fighting Java code fires? Prevent infernos — before code moves into production — with award-winning JProbe from Quest Software.

Quickly pinpoint Java code hot spots with line-level analysis. Discover and debug memory leaks to dramatically improve performance. Automate the task of performing code analysis during off-peak hours. And release applications with confidence, knowing they have been fully tested. JProbe is the proactive solution that gives you higher levels of productivity and end user satisfaction.

Stop Java code performance flare ups — before they start. Improve code quality and increase application efficiency with JProbe.

Watch JProbe in action. View the new product demo at:

www.quest.com/prevent



Automating Governance

For the rules that can be automated, one of the most common approaches is a deployment checkpoint. At deployment time your services are checked against a set of automated rules. These might validate that the services are WS-I-compliant, (increasing their interoperability) and follow further sets of rules that are specific to your organization. This might be that the services use specific predefined schemas, only use certain message transports, etc. The good thing is that this catches non-compliant services before they go into production. The downside is that by the time the service is caught, it's often too late. When it's a choice between meeting a business deadline or following the architecture committee's guidelines, most often the business wins.

The next aspect of automating governance rule validation is applying checks at development time. There are a number of products emerging that can validate the same sets of rules as the "deployment checkpoint" approaches, but do this as a normal and natural part of the development process itself. The advantage of these tools is that they guide the developer down the right path from day one as they build their services, so there's no wasted effort. An added benefit of these tools is that they not only validate that the metadata (such as WSDL) complies with the rules, but they often validate that the content of the messages themselves is also compliant. This includes checks such as whether the messages actually match the WSDL, whether the use of the SOAP protocol is WS-I compliant, etc.

There is a major blind spot in these approaches: they can only validate what they can see. This is where the third aspect of automating governance comes in: runtime governance. There are three different kinds of blind spots in development and deployment time governance products that are addressed by the more advanced runtime governance products.

Blind Spot 1: Service Behavior

While development and deployment time approaches can validate metadata like WSDL and (in some cases) message content, what they can't do is validate that a service behaves according to the rules. For example, does the service properly keep an audit trail in all required cases? Does the service only allow authorized individuals to use it? These are things that can't be validated by development or deployment time governance tools. Even testing tools can't adequately validate that these kinds of rules are enforced in all the requisite cases. In many cases, when these types of rules are implemented in code, the only way to validate that they are properly enforced is by diving deep into the code and evaluating it against a wide series of potential scenarios.

Alternately, you can take advantage of runtime governance tools. These products change governance rules related to behavior from being a coding task to a configuration task. In these products you point and click to declare auditing, security, and other policy behaviors. Moving the enforcement of these rules from a coding task to a configuration task addresses two issues: repeatability—configure these products the same way, and they behave the same way. The same can't be said about custom per-service code. Secondly, since the configuration itself is metadata, validating whether the service meets the governance rules can now be automated, eliminating or at least significantly reducing the chance of human error while simultaneously reducing the time and cost of validation.

Blind Spot 2: Process Awareness

Service Oriented Architectures dramatically change the way you need to think about your production applications.


When a service goes into production, that's not the end – it's just the beginning. The reason is that every time a service is reused, it essentially becomes part of a new application – a new business process – and that business process may have an entirely new set of rules to obey. For instance, a service that's used to store an audit log of information. When the service goes into production you might apply a certain set of governance rules to it – checking those at development and deployment time. Let's say another service – part of a new application – now starts using the audit-log service to store order information as part of an ordering process and that order information includes credit card data. In this case, the service would now be subject to Visa CISP rules *even though the service wasn't changed and wasn't redeployed*. The only thing that changed was how the service was used, and now the set of applicable governance rules changes.

The net result is that you can't assume that development and deployment time governance checks on a service are enough. This is another role where runtime governance comes to the rescue. The most advanced runtime governance products can apply their governance policies not only to individual services, but across entire end-to-end business processes, regardless of when the services were deployed. Since the new business process and thus the new use of the service is what is being deployed, you can validate the policies effectively at

(Continued on page 12)

ABOUT THE AUTHOR

As chief technology officer for Sonic Software, Dan Foody leverages his extensive experience in enterprise systems software toward designing robust and manageable Service Oriented Architectures. Foody's experience with distributed systems technologies including middleware, integration and Web Services, gives him a broad knowledge of the complexities and requirements for managing real-world enterprise software deployments. He is the author of various standards, and has contributed significantly to the OMG standard for COM/CORBA interworking. Recently Foody received InfoWorld's 2005 CTO 25 award. Foody holds a BSEE and MSEE from Cornell University.



The Truth? We had no idea what was causing application performance problems.

Wily can handle the truth. We've helped hundreds of organizations around the world stop the finger-pointing and reliably identify application problems before customers are affected. Using a single solution, you gain control over both your application infrastructure and the customer experience. To learn more about how Wily can help you achieve customer success with your applications, visit truth.wilytech.com.

Get Wily.™

Enterprise Application Management



Where Has My Data Gone?

Accessing Data in a Service Oriented Architecture



BY CLEMENS UTSCHIG AND DOUG CLARKE

With software architecture evolving toward SOA, many projects in this space have encountered challenges associated with accessing data. As has been said, “The way an organization thinks about applications and data must evolve — it must stop thinking about data as a second-class citizen that only supports

specific applications and begin to recognize data as a standalone asset that has both value and utility.”

In today’s world, two different types of data usage can be found: traditional architecturally layered applications and SOA framework-based applications. Naturally, each one comes with

different technological and behavioral characteristics.

On one hand, there are traditional applications — usually designed and written in one language with clear separation between layers, such as enterprise Java with JSP and EJB, or .NET. Communication between layers happens in memory without any intermediate protocol (such as XML). On the other hand, there are SOA frameworks, such as Business Process Execution Language (BPEL) and the concept of the Enterprise Service Bus (ESB), offering everything that looks like a Web Service (exposed through a WSDL) to be orchestrated and to represent itself as a service.

“Organizations should establish their data environments with ‘hubs of specific data families’ that expose data services that comply with industry standards and service contracts. The goal is to create a set of services that becomes the authoritative way to access enterprise data. In this target service-oriented environment, applications and data work together as peers. Thus, both an organization’s business functionality and

ABOUT THE AUTHORS

Clemens Utschig works within Oracle’s SOA Product Management Team responsible for security aspects and cross product integration. Aside from technology, Clemens’ focus is on the project management and consulting aspects that come along with SOA implementations. As a native Austrian, Clemens’ career with Oracle started in Europe at the local consulting services branch — working with customers on J2EE and SOA projects, and founded the local Java community. He is a frequent speaker at conferences evangelizing either technology or the human factor — two key aspects when introducing new concepts and shifts in corporate IT strategy.

clemens.utschig@oracle.com

Doug Clarke is a principal product manager for the Oracle Application Server focused on persistence and Oracle TopLink. Prior to his current role Doug worked as a lead developer, trainer, and professional consultant. Over the past decade his primary focus has been on helping global Fortune 1000 customers integrate relational and non-relational data into their enterprise Java applications. Doug is a frequent speaker at conferences and user groups.

douglas.clarke@oracle.com

data can be leveraged as enterprise assets that are reusable across multiple departments and lines of business.”

In most cases, modular applications already exist and therefore data services need not be built entirely from scratch. This article focuses on aspects of migration and on exposing application functionality for later use in a SOA. It also discusses the pros and cons of the technologies being used for accessing data.

Technologies Used To Implement Data Services

The table below lists common technologies used by applications to obtain data. The question is which of these different implementations of data services should you use in a particular application? There are no hard and fast rules, but this article provides some guidance. Obviously, they can be differentiated through different data formats and different access methods. This article uses these data access technologies to explore the different strengths and weaknesses of data access with SOA enablement.

In a SOA, these technologies are usually composed together, because not all services are implemented in the same technology. This brings up several challenges involving transactional behavior across boundaries, including performance and mass data behavior.

More challenges? So why would you want to introduce a separate set of services in your architecture versus directly accessing a data store? The reasons to consider a separate data service include:

- **Defined interfaces:** Using data services forces you to define contracts that are used between the service and its clients. This is the first step towards abstracting the contract (the interface) from the implementation.
- **Loose coupling/decoupling:** Although a data access layer typically encourages good encapsulation and decoupling of data access functionality, it does not force the issue. Having a separate data service not only forces a well-defined contract but also minimizes implementation details creeping into the client. A client cannot bypass the interface contract

because consumer and provider are not necessarily implemented with the same technology.

- **Reuse:** Using a common data service automatically ensures that all consumers can reuse the same implementation, which leads in the long term to reduced maintenance costs (bug fixes, changes), because code is not duplicated. On the other hand, it requires a well-defined process of change control — because more consumers rely on a functioning piece of code and potential downtime affects more than one user.
- **Flexibility:** Having the implementation completely abstracted from the consumer through a well-defined contract offers greater flexibility. Over time, the implementation technology can change; additional performance enhancements can be introduced; or data stores can be upgraded, migrated, combined, or

divided — all of this without distracting the applications using the data service.

Based on the reasons described to introduce data services into your architecture, the next section takes the challenges arising from the benefits into consideration and maps them against available technologies.

Common Data Challenges in a SOA

These challenges are from the orchestration/business logic point-of-view, such as “How easy is it to access data exposed via JDBC?”

In general, the challenges can be divided into four main groups, each defining a different part of an overall application: access, enrich, distribute, and persist. Usually you start by thinking about how to access a certain data store. Should it be done via handwritten Java code that embeds SQL into JDBC calls? Should the Java Connector

Physical Characteristics	J2CA (Java 2 Connector Architecture)	Native JDBC	Remote (EJB)	Web Service
Data Format	Record or Custom structure	Result Set	Serializable Java Objects	SOAP/XML
Access Method	J2CA is part of the Java EE specification and is used within a Java EE Container	In both Java SE and Java EE applications	Can be accessed from Java SE and Java EE applications	From everywhere and even with different implementation languages (Java/C++/VB/.NET/ASP.NET, etc.)
Transactions	Native J2CA is transaction aware and can participate in a two-phase commit	JDBC provides native access to the database therefore the transactional behavior has to be controlled from within the application directly	EJB is by definition transaction-aware. This means a set of EJB method calls can participate in one transaction (and therefore can be rolled back)	Support is only in parts, mostly by using WS-TX (Web Service transactions). Spanning multiple calls into one transaction can be achieved with WS-Coordination. However, in a case with many loosely coupled interactions, the concept of compensation transactions is more common. To use this concept every action has to provide a counteraction, e.g., bookHotel() and cancelHotel()

Table 1:

ACCESSING DATA

Architecture (J2CA) be used to connect to a foreign data source? Is the client Java or a .NET application — and therefore can a native protocol be used or not?

After considering the access options, the next step is to validate whether aggregation of data across boundaries is necessary, whether a high data load is expected, and where the data comes from. All of these options are considered in the discussion of Challenge 2 (enrich, cleanse, and aggregate data).

Another major challenge involves which and how many users plan to consume a service. In particular, the importance of interoperability should not be underestimated if data is being pulled from or pushed to a consumer (such as business-activity monitoring [BAM]). This is described in the discussion of Challenge 3 (distribute data).

Last but not least, there is the challenge of writing data back and guaranteeing consistency across multiple calls to a service as well as across boundaries. In SOA, use cases are generally implemented across technological and service boundaries as they are orchestrated into a process. These questions are covered in the discussion of Challenge 4 (persisting data).

Challenge 1: Access Data

Abstraction of data - A domain model is abstracted from the underlying data store. This model should be designed according to the requirements of the client applications. It could mirror the underlying data store or provide a rich abstraction. In either case, the consuming application is decoupled from the physical storage by this model.

In reality this means that a customer object used in the application usually differs from the underlying data stores and their view of data, such as object-oriented versus normalized.

Speed considerations - Allow for efficient retrieval of domain objects based on exposed operations. This is basically the exposure of queries the application needs. It also provides a natural boundary for testing and for the introduction of mock data in test scenarios. Because the next generation of applications and business processes

Accessing Data	Technology Choice			
Technology Characteristic	J2CA (Java 2 Connector Architecture)	Native JDBC	Remote (EJB)	Web Services
Ease of Development	Hard. Developers have to understand both the source system and the target implementation language	Moderately easy to develop and run any SQL possible. Hard aspects are mapping types. Knowledge of SQL is required. Complex databases can involve complex JDBC/SQL coding	With today's generation tools and EJB 3.0's Java Persistence API (JPA), this is easy. Mapping to the DB tables doesn't cause much pain at all	Web Services can be exposed from almost any object or structure today. Very easy to develop
Reuse (Ease of)	Yes, but the adapter instance only – no business logic	Moderate reuse of SQL statements, other logic not written in SQL is tied to the implementation	Remote EJBs are made for remoting and can be reused very easily	Web Services can be accessed from various technologies, not just Java
Security	Yes, the connector is secured by means of Java EE	The database is secured; security considerations are entirely on the database side	Due to the Java EE security infrastructure this is possible but implementation is hard	Web Service standards offer sophisticated security models for both authorization and authentication. However, it is best not to embed security policies in the app code – it is better to capture and execute it in a Web Services management and security solution
Performance	Yes, due to native in-memory access (application to adapter communication)	If SQL is slow, the whole app is slow. Use if the aggregation and mapping needs to be fast – and can be done through SQL	EJBs can be accessed locally and remotely. The native protocol is RMI. Many querying and data retrieval optimizations available	Performance is affected by serialization to and from XML on top of the underlying implementation costs
Ease of Change	The adapter is portable, but the business logic resides in the app	Client code – therefore it is hard. If logic resides in the database, it is hard to port. Changes to schema or queries can be expensive across all embedded SQL code	Operations are within the Java EE container, such as administration and deployment. Mapping changes can be isolated from the client	Because these services are locked to the interface but implementation technology can change, change is easy. Nevertheless, it has to follow a process

Table 2: How the technology behaves when being used to access data

are composites that leverage these services, data access abstraction must be considered a key aspect of a service-oriented testing strategy

Challenge 2: Enrich, Cleanse, Aggregate Data

In modern applications, data usually does not come just from one place and get presented directly to consumers. It often needs to be formatted (e.g., a date), aggreg-

gated or enriched, or cleaned.

For example, a big part of an employee record might come from the HR system but the employee's vacation time is stored in some other place requiring a merge of those two sources into one.

There are several ways aggregation or enrichment can be implemented. At the most basic level, it can be done natively in the application no matter what the technology. However, if the underlying data model

Aggregating Data	Technology Choice			
Technology Characteristic	J2CA (Java 2 Connector Architecture)	Native JDBC	Remote (EJB)	Web Services
Ease of Development	Requires high skill levels	Enriching and aggregation of data can be achieved via SQL, but if aggregation happens with different sources, it is more difficult and requires significant custom coding	EJBs can be grouped and “orchestrated” through Session Facades – which allows for data enrichment and other business logic, but this requires significant coding effort	Aggregating Web Services is really easy, because of orchestration languages, such as BPEL, offering the ability to put them together into a business process. Also true for ESB
Performance	In container, in memory—good	Good performance can be achieved through in-database operations. If aggregation happens outside of the database, performance declines	Dependent upon the underlying data access. Potentially hard and performance-killing. Aggregation has to happen in the model	Performance really depends on the implementation of the service. The amount of serialization is especially critical to performance. In this case every element and byte counts
Ease of Change	Changing the logic results in high maintenance and effort, because it affects the whole application	Same as native JDBC	If separation is done correctly, only one part (one EJB) is affected	Orchestration can be changed, so even if the change affects the whole service, it's never the whole application

Table 3: How the technology behaves for aggregated data

changes (or just a column name changes), several places need to be updated, which can be time-consuming. This leads to the use of object-relational mapping solutions because the domain model is easily shared and the mapping can be changed just in one place. On the more sophisticated side the data service implementation could access multiple data sources using native results to compose the common domain model.

When multiple data sources return a variety of XML documents that comprise the common domain model then XML aggregation is the answer. Due to the evolution of orchestration and XML as a common data format, BPEL is the perfect tool for aggregating or enriching data that is already in XML and later serving it to consumers. Although for large payloads and results, using an Enterprise Service Bus might be a better choice.

Challenge 3: Distribute Data

When all components and layers reside within one application, access is native and in memory. In a SOA, however, services run in a decentralized manner and in different places, introducing the challenge of different protocols and even greater need for solid error handling. What happens if a service is down or not accessible — how will the application be affected? The use of BPEL offers a great way of composing these services into one process or composite service, which can be leveraged later. Using this approach brings more freedom to the service's consumers but also introduces another layer that needs to be maintained.

One purpose of this is data pulled into several UI technologies, such as a portal or BAM to monitor performance.

For example, a supervisor wants to see the performance of employees in real-time and wants to be able to take corrective actions (such as rerouting a request to

other teams). In this case, the data from the business process (and its services) can be pulled out into a BAM instance to create a real-time dashboard.

Note: The technology mapping of this section has already been covered in the Access table.

Challenge 4: Persisting Data

A data access solution must provide proper transactional support allowing the application to apply changes to the data stores through the exposed operations and the supplied changes to the domain model, ensuring that the proper transactional semantics are obeyed, based on the data store's requirements and implementing any necessary concurrency protection (locking).

Especially with SOA, more than one application will use a certain service at the same time, so transaction protection needs to be considered. One possibility is that each operation can be reverted. Another option is to sacrifice loose coupling to allow clients to use transactional behavior.

Evolving a Multi tier Application into a Reusable Set of Services

The goal for the next generation of applications is to morph them smoothly into a SOA — and not force them to be completely rewritten. If a clean separation of layers was introduced earlier, this may sound harder than it actually is. Let's review two possible morphing approaches:

1. Traditional application (architectural model)

Following all the design patterns, a clean separation of concerns is desirable — especially moving forward to a SOA. In the following model, the application (representing the business logic) talks to the persistence via a defined set of interfaces and transports its data through Plain Old Java Objects (POJOs).

2. Morphing into a hybrid model

Expose just the needed artifacts and functions into service interfaces (a.k.a. service enablement). These can be further used in BPEL to orchestrate complex business processes. The application, on the other hand,

ACCESSING DATA

Persisting Data	Technology Choice			
Technology Characteristic	J2CA (Java 2 Connector Architecture)	Native JDBC	Remote (EJB)	Web Services
Ease of Change	Change is hard, because the record structure differs adapter to adapter, even if the interfaces are standard	Extremely hard, because the mapping of columns is coded somewhere in the application	While the bindings are embedded in the application the change is much simpler than a hand-coded JDBC/SQL layer	Easy, because with a defined contract, the implementation can change without affecting the consuming application
Performance	As mentioned, J2CA adapters can participate in global transactions and provide the means for a two-phase commit	When the whole application uses only native JDBC access, it can control the commit cycles and the locking of resources around them. However, in a SOA, commits are atomic, therefore optimistic locking should be used	EJBs provide transactional capabilities and can participate in container transactions. However, when exposed as a Web Service, commits become atomic, meaning after each completed method commit is called implicitly therefore optimistic locking should be used. When you use them in a business process, you should foresee compensation logic	Performance really depends on the implementation of the service. The amount of serialization is especially critical to performance. In this case every element and byte counts
Ease of Change	Changing the logic results in high maintenance and effort, because it affects the whole application	Same as native JDBC	If separation is done correctly, only one part (one EJB) is affected	Orchestration can be changed, so even if the change affects the whole service, it's never the whole application

Table 4: How the technology behaves in the context of persisting data

still uses the native, in-memory approach and is not affected at all. The model stays the same, and just certain pieces are exposed (the same interfaces!).

Integrating Data Through Services: A Use Case

Having discussed the pros and cons of various access technologies, such as

JDBC, J2CA, Web Services, and Enterprise JavaBeans (EJB) as well as the challenges of accessing data, it's time to apply the information gained to a use case.

A large company runs its inventory system as a mainframe application that is maintained and updated through daily batch jobs. Due to the rising demand of a new front-end system, the company decided to develop a custom Java EE application, with a Java Server Faces front-end, capable of serving multiple clients but holding the mission-critical information in the back-end. Between those two systems, new items should be exchanged and enriched with external data from a supplier. The service for retrieving this information is offered through a secured Web Service. The architecture is laid out as follows:

Connecting these applications – all using different technologies – serves as a perfect example to illustrate choosing the right data access, aggregation, and persistence approach to build a best-of-breed solution.

Applying the technologies discussed about, J2CA connectors, which provide native access to the underlying technology, can be used for the mainframe. Most of these adapters also provide a WSDL interface to the outside world, describing their exposed services. Each time a new item is triggered, the adapter fires an event.

The Java EE application consists of a data access layer built on top of EJB 3.0 to ensure flexibility in the mapping between domain objects and the actual database

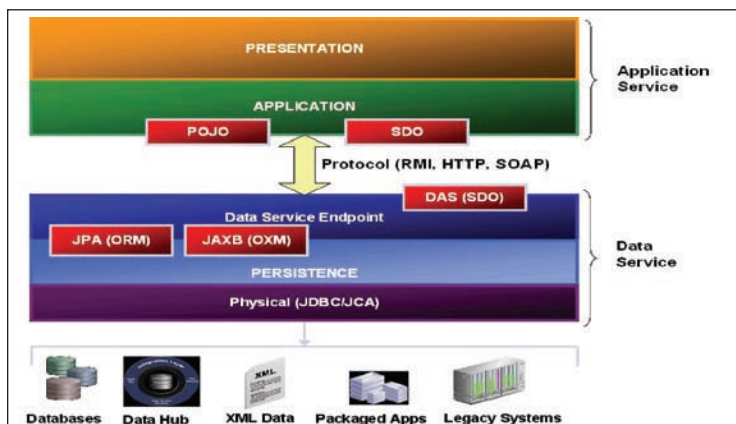


Figure 1: Complete separation of logic, with access only through remote protocols

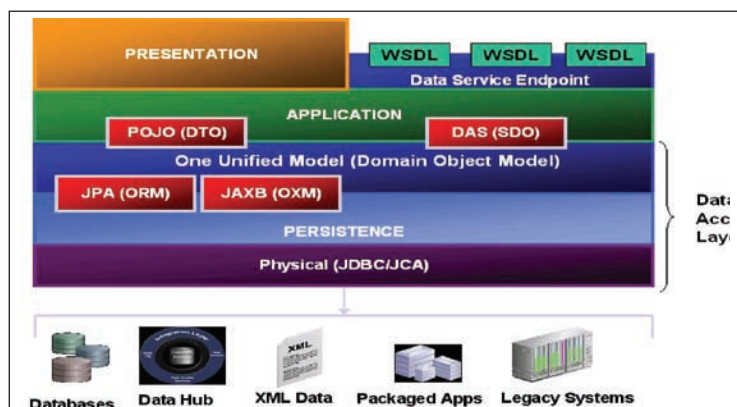


Figure 2: Layered model in which business logic is used locally as well as exposed for remote access



IT'S IN THERE SOMEWHERE

MAKE ANSWERS TO PERFORMANCE PROBLEMS COME TO YOU.



OPNET Panorama offers powerful analytics for rapid troubleshooting of complex J2EE/.NET applications. Panorama quickly identifies how application, web, and data-base servers are impacting end-to-end performance. With Panorama, you can pinpoint the source of a problem, so time and money aren't spent in the wrong places.

The most successful organizations in the world rely on OPNET's advanced analytics for networks, servers, and applications.

www.opnet.com/pinpoint

OPNET
Making Networks and Applications Perform™

OPNET Technologies, Inc. 7255 Woodmont Avenue, Bethesda, Maryland 20814 phone: (240) 497-3000 • e-mail: info@opnet.com • NASDAQ: OPNT

© 2006 OPNET Technologies, Inc. All rights reserved. OPNET is a registered trademark of OPNET Technologies, Inc.

(Continued from page 4)



schema. These EJBs can either be exposed as real Web Services or offer just a WSDL interface but require native access.

In this case, an ESB can be used to provide the running infrastructure for orchestrating an overall flexible process. It not only contains rules for routing and aggregation but also offers error handling and ensures a high degree of data access and persistence performance by using native protocols.

Outlined Process Flow (ESB System Diagram)

As this example shows, an ESB can support the loosely coupled principles of a SOA but also address common data access challenges.


Conclusion

A data service is a means of decoupling and encapsulating the access to one or more data stores. This concept offers an approach to sharing common functionality across multiple client applications or services. Its physical separation allows much greater flexibility in implementation and future independent evolution while guaranteeing that the consumer and the data service negotiate on a contract. The benefits of this approach include the ability to transparently aggregate data and even completely change data stores without requiring changes to its consumers. Additionally, it allows for a data service that may have once been used only in a fixed set of applications to be used within a business

process as a more dynamic service orchestrated with declarative processes.

Each technology that has been discussed has pros and cons (starting with performance and ending with transactionality). We believe that it is important to keep those in mind and that the more data-rich the application is (the more mass data it has) the more a native approach is appropriate. On the other hand, the more loose coupling is targeted, the more the approach should lean toward a Web Service-based architecture. In this sense, using BPEL seems like a great way to enable data sources into services and allow performance and, to some extent, transactionality.

The key to a successful transformation is a solid understanding of the purpose of the data service, including the pros and cons of each technology used. The use of proven data access frameworks makes exposing data as services simple while not sacrificing performance. Good performance is essential in multi-layered applications, because it is crucial to the user experience. The best SOA is worth nothing if the users are sick of using the consuming applications due to their poor performance.

Although having shared data services appears to be an obvious solution for shared access to the same enterprise data stores, it does not come without costs. As in all design and architectural abstractions, the benefits must be carefully weighed against the costs and challenges. A vision of the enterprise's SOA strategy must also be taken into consideration when deciding when and where to use data services. This article has attempted to alleviate the difficulty in addressing the challenges of making these decisions in a SOA by describing and assessing the characteristics of data accessibility. 

Reference:

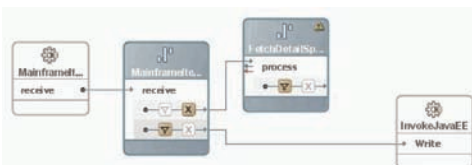
<http://webservices.sys-con.com/read/233667.htm>

business-process deployment time. In contrast, without awareness of a new context of use, the business process, you'd be forced to re-analyze each service that's already in production the moment another application is deployed – a very complex and time-consuming challenge.

Blind Spot 3: Rogue Services

Up until now, we've gone with the assumption that the governance review process is aware of all of the services and uses of services that are going into production. But, is this a realistic assumption? It turns out that in many cases it's not. If a service or service-use gets into production and it didn't go through the proper approval process, you have what's called a rogue service. Rogue services are organizational risks because you just don't know whether they're in compliance or not. It doesn't matter how well you tried to follow your process – if a service gets into production and it's not auditing financial data (and so isn't SOX-compliant) someone might go to jail. The SEC doesn't give you amnesty because you *tried* to follow your process.

Rarely are rogue services the result of malicious acts. Most people in an organization don't try to bypass the approval process – it can happen for a lot of innocent reasons. For example, let's say you're deploying a packaged application or an application built by a third-party outsourcer – you might not be aware of all of the services contained in this application. Even when you are, sometimes there are just too many to fully evaluate – SAP, for example, has hundreds of services ready to use out-of-the-box. A second case might be a service that was built purely for internal use in an application and so wasn't subject to the approval process – but someone in another application gets a hold of it and starts using it. When you talk about rogue service use, the set of cases where this can occur grows even longer. One organization relayed a story of how they had built a service that had five authorized consumers (each of which had been



Subscribe Today!

— INCLUDES —
FREE
DIGITAL EDITION!
(WITH PAID SUBSCRIPTION)
GET YOUR ACCESS CODE
INSTANTLY!

issued a special consumer key so that the service owner could track them), but it turned out there were 34 different consumers. What happened was one of the five authorized consumers had built the use of the service into a jar file. The jar file embedded the consumer key for simplicity. Twenty-nine other project teams reused this jar file without knowing that it happened to use an external service – so they unwittingly reused the service. And these service uses didn't get approved; they were rogue service uses.

How did this organization find out about these other uses? It turned out that, to find a performance problem with their service they deployed a runtime governance product (one of the common capabilities of runtime governance products is service-level measurement) – since they thought there were only five consumers, they didn't understand why it wasn't performing as expected. The runtime governance product they deployed could also automatically discover new services and new service consumers. This product automatically discovered all 34 consumers. By interfacing with the company's registry of approved services, the product determined that 29 of these consumers were actually rogue consumers and immediately flagged these for approval. The most advanced runtime governance products can even automatically quarantine rogue services and service uses until they're approved – eliminating the risk of rogue services.

Bringing It All Together

To implement a complete approach to SOA governance, you have to consider the roles of development, deployment, and runtime governance. Taking a holistic view of governance across the lifecycle will automate as much of the governance burden as possible, while providing a backstop to catch the rogue services and service uses that your human-centric processes don't catch. Of course, there's no perfect solution – the human element still plays a key role. To reduce risk, you have to reduce the complexity of the manual processes – so remember to think strategically about which rules are really necessary, and which are just “nice to have.”



M A Y 2 0 0 6



*The major infosecurity issues of the day...
identity theft, cyber-terrorism, encryption,
perimeter defense, and more come to the
forefront in ISSJ the storage and security
magazine targeted at IT professionals,
managers, and decision makers*

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹

ONE YEAR
12 ISSUES

www.ISSJournal.com
or 1-888-303-5282

SYS-CON
MEDIA

The World's Leading i-Technology Publisher

Rich Internet Applications: AJAX,

www.AjaxWorldExpo.com

AJAXWORLDTM

CONFERENCE & EXPO

SANTA CLARA SILICON VALLEY

**SYS-CON Events is proud to announce the first-ever
AjaxWorld Conference & Expo 2006!**

**The world-beating Conference program will provide developers and IT managers alike
with comprehensive information and insight into the biggest paradigm shift in website design,
development, and deployment since the invention of the World Wide Web itself a decade ago.**

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this uniquely timely conference – especially the web designers and developers building those experiences, and those who manage them.

CALL FOR PAPERS NOW OPEN!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested learn about a wide range of RIA topics that can help them achieve business value.

Flash, Web 2.0 and Beyond...

REGISTER TODAY AND SAVE!

→ **October 3-4, 2006**

→ **Santa Clara Convention Center**
Hyatt Regency Silicon Valley
Santa Clara, CA

→ **To Register**

Call 201-802-3020 or
Visit www.AjaxWorldExpo.com

→ **May 7-8, 2007**

First International AjaxWorld Europe
Amsterdam, Netherlands

“Over the two information-packed days, delegates will receive four days’ worth of education!”

Early Bird*

(Register Before August 31, 2006)
..... **\$1,495****
See website or call for group discounts

Special Discounts*

(Register a Second Person)
..... **\$1,395****
See website or call for group discounts

(5 Delegates from same Company)
..... **\$1,295/ea.****
See website or call for group discounts

On-Demand Online Access

(Any Event)
..... **\$695**

*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

“It Was The Best AJAX Education Opportunity Anywhere in the World!” —John Hamilton

Topics Include...

Themes:

- > Improving Web-based Customer
- > Interaction
- > AJAX for the Enterprise
- > RIA Best Practices
- > Web 2.0 – Why Does It Matter?
- > Emerging Standards
- > Open Source RIA Libraries
- > Leveraging Streaming Video

Technologies:

- > AJAX
- > The Flash Platform
- > The Flex 2 Framework & Flex Builder 2
- > Microsoft's approaches:
ASP.NET, Atlas, XAML with Avalon
- > JackBe, openLaszlo
- > JavaServer Faces and AJAX
- > Nexaweb
- > TIBCO General Interface

Verticals:

- > Education
- > Transport
- > Retail
- > Entertainment
- > Financial Sector
- > Homeland Security

GROUP DISCOUNTS AVAILABLE:
— 5 Delegates from Same Company —
for only \$995 (each)
— Register a Second Person —
for only \$1195

**Hurry! Limited Seating
This Conference Will Sell-Out!**



LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

Receive **FREE**
WebCast Archives
of Entire Conference!

The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.



► This on-demand archives set is sold separately for \$995



For more great events visit www.EVENTS.SYS-CON.com

VISIT WWW.AJAXWORLDEXPO.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

Forgetting something?



You are,
if you're not using
PowerGen
for your PowerBuilder development.

PowerGen removes all obstacles to performing fully automated source control and build procedures for your PowerBuilder applications.

Visit ecrane.com to see what you're missing.