Web Services Roadmap

*A **CBDI** Report Series-*
*Guiding the **Transition** to **Web Services** and **SOA***

Introducing Web Services and Service Oriented Architecture (SOA) requires change right across the organization for both end user enterprises and the software industry. These changes, technical, organizational, process and project related, will happen over a period of time and with varying levels of coordination. Some of the changes will be obvious, others need to be identified; many will need organization specific solutions.

To assist organizations making these changes CBDI has created the Web Services Roadmap report series to provide practice guidance in this area.

Inside…

- **Web Services Roadmap Planning Framework**
- **A Web Services Maturity Model**
- **ROI – The Costs and Benefits of Web Services and Service Oriented Architecture**
- **The Web Services Protocol Stack**
- **Assembling the Web Service Infrastructure**
- **Moving to SOA**
- **ISVs and Packaged Application Vendors Start Here**
- **Applying Web Services**
- **Real World Migration of Development Projects to SOA**
- **Web Services Roadmap for the On Demand Business**
- **Microsoft in Transition - Delivering a Less Complex Service Oriented Platform**
- **Practical Support for Separate Supplier and Consumer Activity**

About The Web Services Roadmap

Introduction

Introducing Web Services and Service Oriented Architecture (SOA) requires change right across the organization for both end user enterprises and the software industry. These changes, technical, organizational, process and project related, will happen over a period of time and with varying levels of coordination. Some of the changes will be obvious, others need to be identified; many will need organization specific solutions.

To assist organizations making these changes CBDI has created the Web Services Roadmap to provide practice guidance in this area. It is organized into six topic streams, providing a clustering around critical activities as follows:

Plan & Manage. The activities involved in managing the transition to a more federated environment, enabled by SOA; the development and coordination of common policies and practices between the parts of the federation.

Infrastructure. Guidance on the strategies, activities and timing involved in transitioning existing infrastructures.

Architecture. Architecture is a major issue. As an organization moves forward it is looking to integrate Web Services into core business processes, to progressively establish the service as the unit of reuse across an organization. The Service Oriented Architecture is a key strategy to achieve pervasive shared business services.

Process. In the service oriented world, the management life cycle changes in a profound manner. Organizations need to alter their application acquisition and delivery processes to reflect this.

Projects. As an organization matures in its use of services, the project profiles change considerably.

Vendor Strategies. Analysis and assessment of the strategies and capabilities of the leading vendors in the Web Services and SOA market.

Web Services Roadmap On-line

The CBDI Web Services Roadmap continues to be extended and updated on-line. You will also find the latest related news and an opportunity to leave feedback and discuss issues with your peers. This is provided at no charge, and no registration is required. CBDI Corporate Subscribers are also able to download all the roadmap materials as Microsoft PowerPoint presentations to further assist them in communicating their plans to their colleagues.

Visit http://roadmap.cbdiforum.com

Authors

David Sprott – david.sprott@cbdiforum.com

Lawrence Wilkes – lawrence.wilkes@cbdiforum.com

Richard Veryard – richard.veryard@cbdiforum.com

Jonathan Stephenson – jonathan.stephenson@cbdiforum.com

The CBDI Web Services Roadmap is Sponsored by

COMPUWARE. IBM. Microsoft® SELECT™ BUSINESS SOLUTIONS

Audience

This report assumes readers have a basic understanding of Web Services and SOA.

It is aimed at project managers and lead developers, development managers, CTO's, technical and application architects, business analysts, senior consultants and product strategists and managers.

It is equally applicable to both end user organizations and IT vendors.

Using this Report

This report has not been designed to be read sequentially from cover to cover like a novel. Rather, it is more of a reference work that readers can turn to for guidance on introducing Web Services and SOA into their organization.

Copyright

Contents

Web Services Roadmap Planning Framework 4
- **Introduction 4**
- **The Basic Roadmap Model 4**
- **The Roadmap Framework 5**
- **Plan & Manage Stream 6**
- **Infrastructure Stream 11**
- **Architecture Stream 14**
- **Process Stream 18**
- **Projects 25**

A Web Services Maturity Model 27
- **A Strategic Perspective for Technology and Business Planning 27**
- **Introduction 27**
- **Technology Maturity 27**
- **Business Maturity 28**
- **Four Phases 28**
- **Timing 31**
- **Summary 32**

ROI – The Costs and Benefits of Web Services and Service Oriented Architecture 33
- **Why Web Services? 33**
- **Why SOA? 37**
- **The Costs of Web Services and SOA 37**
- **The Cost of SOA 38**

The Web Services Protocol Stack 40
- **Web Services Architecture 40**
- **CBDI Assessment 41**
- **Roadmap Actions 44**

Assembling the Web Service Infrastructure 45
- **Introduction 45**
- **Web Service Infrastructure Architecture 47**
- **Web Service Management 49**
- **Stages of Web Service Infrastructure Deployment 49**
- **Considerations and Inhibitors 51**
- **Needs of the Large Enterprise 51**

Moving to SOA 53
- **Introduction 53**
- **It's About Better Integration 54**
- **Creating the Foundation 55**

- The Business Service Bus 56
- A Framework for SOA 57
- Summary 60

ISVs and Packaged Application Vendors Start Here 61
- Introduction 61
- ISV Deltas 61
- ISV Deployment Patterns 63
- Competitive Threats 64
- Web Services Adoption Steps 64

Applying Web Services 66
- Introduction 66
- Conversion Scenarios – The First Steps towards Web Services 66
- Where to Apply Web Services 70
- Exploitation Scenarios – Capitalizing on Web Services 72
- Application Policy 75
- Managing Participation 76

Real World Migration of Development Projects to SOA 77
- Introduction 77
- Using MDA to Move to SOA 77
- Using OptimalJ to Move to SOA 79
- JCA Integration Model 81
- Web Services Support 82
- Model-based Pattern-Driven Code Generation 82
- Summary 83

Web Services Roadmap for the On Demand Business 84
- On Demand Business 84
- The IS Department Challenge 85
- Web Services are Key 85
- Components of Web Services Success 86
- Delivering Best Practices 88
- Web Services Roadmap 90
- Summary 94

Microsoft in Transition - Delivering a Less Complex Service Oriented Platform 95
- Technology Platform and Complexity Reduction 95
- Architecture – Inside and Out 96
- Service Oriented Architecture 96
- The Mission to Jupiter 97
- The Roadmap to SOA 97
- Mobility and Smart Devices 98
- Summary 99

Practical Support for Separate Supplier and Consumer Activity 100
- Introduction 100
- Supply Manage and Consume (SMaC) 100
- Four Routes to SOA 101
- New Solution 102
- OO & Micro-Componentry 103
- Case Studies in Enterprise CBD 104
- Legacy & COTS 105
- Select Business Solutions 109

Web Services Roadmap Planning Framework

Abstract: Introducing Web Services requires some changes right across the organization. These changes, technical, organizational, process and project related, will happen over a period of time and with varying levels of coordination. Some of the changes will be obvious, others need to be identified; many will need organization specific solutions. In this report we provide a planning framework that can be used as either a starting point or as review input.

Introduction

A Roadmap is typically used to plan a journey. The roadmap is a general purpose device that allows route planning where the start point is known, and there are potentially many alternative routes that could be taken to reach a known endpoint. Whilst we won't stretch the metaphor too far, the concept of Roadmap planning for IT systems has been widely used, as a device to coordinate many disparate activities which are often widely dispersed in terms of time, geography and accountability. Hence the Roadmap approach seems highly appropriate for Web Services introduction.

In this report we will consider the requirements of the typical enterprise. Many other types of organization such as ISV's, Systems Integrators, Intermediaries and Service Platform Providers may also benefit from a Roadmap, and the base enterprise model may be a useful context (as a supplier, consumer or intermediary) for planning.

In “A Web Services Maturity Model”, we introduce the concept of Phases of Web Service related activity. We suggested that whilst there will always be exceptions, there is a mainstream adoption profile which follows a path of early learning; integration, reengineering and finally maturity. In this report we will reuse and build upon this basic phase structure, identifying the relevant policies, capabilities and tasks that may need to be put in place in each of the phases.

The Basic Roadmap Model

In Figure 1 we introduce the basic Roadmap model. The x axis repeats the phases discussed above. On the y axis we provide a structure for thinking about timing and interdependency which are simply clustering mechanisms, which specifically remove any organizational relevance. We refer to these as streams.

The basic Roadmap model has many potential uses. These might include:

- Making strategic choices, for example identifying the states that an organization may plan to achieve during each Phase.
- Guiding disparate activity across an organization, providing an organization independent model that many parties can easily subscribe to.
- communicating cross organizational activity in an organization independent manner

We have found that a high level view of this type can be very useful in sorting out the overall directions. So in this illustration we have charted an example enterprise that follows a fairly conventional path, which many will recognize. Early learning is undertaken in a deliberately uncoordinated manner, with the minimum of formality. The integration phase then requires certain matters to be managed in order to achieve sensible levels of consistency, which will hopefully reduce overall cost to the organization as well as establish common foundations for matters such as classification, core infrastructure services etc. The reengineering phase is then focused on creating enterprise level services and platforms, and maturity is all about convergence of the IT Services and the business products and processes.

Plan & Manage	N/A	Fund Common Infrastructure	Enterprise level Funding Service Based Org	Federated Management
Infrastructure	Basic security policies	Common Platform and Interop Bus	Common management	Virtual Business
Architecture	N/A	SOA Governance	Business Service Bus	Everything is a service
Process	WS- I Profile Coordination	SLA and Classification	Collaborative Framework	Product Line
Projects	Experimental	Project services	Enterprise services	Services are Products
	Early Learning	**Integration**	**Reengineering**	**Maturity**

Figure 1 - - Example of Enterprise Roadmap Strategy

We have based this particular model on a very common pattern which we might refer to as large distributed enterprise. The base model may form a good basis for developing variants, that would be suitable for alternative patterns such as integrated enterprise; virtual enterprise; Web Service is a product; and so on.

The Roadmap Framework

In this section and the remainder of this paper we focus on a more detailed guide to Roadmap activity. We provide a first level of decomposition of the streams, with topic areas and deliverables, together with rough indications of Phase applicability. We stress that this guide is intended as a starting point and or review input, in order to assist organizations to customize and manage their activities and plans.

Plan & Manage Stream

Topic Area	Deliverables	WS Maturity Phase			
		EL	INT	RE	MAT
Service strategy	Agreed roadmap - Organizational consensus on the overall plan for transitioning business and technology environments, synchronizing relative maturity of capabilities and applications		Y	Y	Y
Business and Technical Justification	Identified Business Opportunities Support to Technical Strategy		Y	Y	
Coordination, Communications & Policy Management	Identified areas for cross organizational coordination Mechanisms for cross organizational policy setting, projects support, decision making	Y			
Funding & Charging	Funding Strategy & Policy		Y	Y	
Organizational change	Roles and responsibilities		Y	Y	
Classification systems	Common classification policy and schemas	Y	Y		
Service Level Management	SLA policies - levels and processes		Y	Y	
Monitoring and reporting	Management policies and systems		Y	Y	Y
Marketing	Promote common services		Y	Y	
Services as business products	Policies and plans for implementing a Product line or equivalent process	Y	Y	Y	
Security and Trust	Business and Technical Security Policies	Y	Y	Y	Y

Table 1 - The CBDI Roadmap Framework – Plan and Manage Stream

The transition to a more federated environment, enabled by SOA, requires the development and coordination of common policies and practices between the parts of the federation. The management stream provides a clustering of topics and deliverables required to establish these.

Service strategy

Generally the requirement for, and understanding of a service strategy will surface after only after early learning experience has demonstrated the need for common services, classifications and agreements. However our observation is that the strategic nature of service orientation as it applies to an enterprise is likely to develop progressively over time. An important factor in this is business management understanding. In our survey, carried out in early 2003[1], there was very low awareness outside of technical management, and this will take time to permeate outside the IT world, given the relatively slow take-up of Web Services and the dismal economic climate which militates against IT led investment. Also from the IT perspective in the archetypal enterprise, there is much work to be done in creating a critical mass of existing and legacy applications in an SOA, before real cross enterprise business advantage will be perceived.

1 CBDI Web Services Usage Survey, February/March 2003
http://www.cbdiforum.com/bronze/webserv_usage/webserv_usage.php3

The roadmap concept is therefore recommended as a living deliverable, which will in the early stages reflect the technical nature of the tasks at hand, but will allow technical managers to communicate in understandable terms how and when the IT investment will support new business models.

Business and Technical Justification

A key part of the roadmap process is guiding the use of new technologies appropriately. Many organizations have moved slowly with Web Services because they do not have clear justification for implementing new protocols, and particularly the necessary infrastructure and management environment, when existing arrangements work entirely satisfactorily.

In our research on ROI and cost justification[2] we have primarily focused on benefits to specific and individual applications in terms of productivity, quality, ease of integration and enablement of new business models. In the context of Roadmap planning there is the additional opportunity to take an enterprise wide view of the infrastructure investment and returns.

Coordination, Communications & Policy Management

Many organizations do not have mechanisms for enterprise wide policy setting. More than a few enterprises tell us they deliberately operate different technical and application sourcing policies in order to segregate business divisions. Those organizations that have already been developing intra and inter company document exchange based on XML will have established organizational mechanisms to manage document content semantics and schemas, and also transport, document security and guaranteed delivery arrangements. With Web Services the cross and inter enterprise coordination requirements expand to include service level agreements, contract agreements, process choreography and service management.

Funding & Charging

You may think that charging automatically moves to a "per service" basis, simply because it's made easier by a) the business transaction relevance and b) the management and monitoring systems that make per service billing easy. However think before you charge, because all of the time honored intricacies of charging apply here also, and free, partial per service and many other arrangements may apply. It's all about behavior modification through the pricing mechanism.

The most important principle is that many Web Services will not be chargeable at all - because the Web Service is an intrinsic component of a business service or transaction. So the purchaser of a book from Amazon pays for the book, which in some manner subsumes the cost of providing the service. But clearly there will be some level of internal accounting within Amazon that accounts for product costing. We can rationalize this as two different classes of Consumer charge

- Usage charging, charging someone for using the service, and
- Cost recovery, charging for the execution of the service, which is often most relevant when the service is provided to the end user at no charge.

We discussed usage charging in our report Component Pricing[3]. We suggested that at this early stage, we are inclined to the opinion that the well known Boston Grid will be the dominant influence in macro pricing strategy. Subscription-based pricing seems particularly appropriate for mature products – which may represent “cash cows” for the suppliers – where the users usually know what they’re getting. “Cash cow” is one of the four stages of the product lifecycle, known as the Boston Grid.

Other stages of the product lifecycle probably need other pricing schemes, as shown in the following table.

2 CBDI Report - Inside Every Web Service is a Benefit Struggling to Get Out
http://www.cbdiforum.com/bronze/inside_web.php3
3 CBDI Journal Report - Component Pricing, July/August 2001
http://www.cbdiforum.com/secure/interact/2001-08/comp_pricing.php3

Product stage	Market dynamics	Market share	Objectives	Pricing model
Problem child	*growing*	*small*	Develop product. Encourage experimentation	Low unit pricing.
Rising star	*growing*	*large*	Maintain market share. Encourage customer roll-out,	Site licenses. Volume discounts.
Cash cow	*stable*	*large*	Maintain long-term revenues from product	Subscription-based licence
Dying dog	*stable*	*small*	Transfer customers onto alternative products	

Table 2 - Boston Grid

In the report we went on to say - Although subscription pricing remains rare for software components, it has sometimes been predicted as the dominant model for Web Services. However, we believe it would be incorrect to assume that Web Services will automatically be priced in this manner. Whilst consumer Web Services (B2C) may largely adopt subscription pricing, business (B2B) Web Services are much more likely to be an integral part of a product (bundled) or priced in some manner that modifies behavior.

Organizational Change

Service architectures introduce clear separation, with a high level of formality between provider and consumer. In the early learning stages things will remain pretty much unchanged. But as soon as services are better understood there's massive potential for productivity and quality gain.

Many organizations that embraced component based development (CBD) implemented significant levels of separation between the supplier and consumer of a component. This base concept is fundamental to the service world, but for services we have the addition dimension of design time and run time reuse. In Figure 2 we show a real world perspective where we have new parties involved and changed responsibilities in the process:

- **Developer -** yes of course the developer still exists, BUT now the developer is providing to the provisioner who undertakes the publishing tasks.
- **Process Owner -** the process owner's view is now entirely service based. The process owner specifies, acquires, assembles, tests and deploys services.
- **Provisioner -** the provisioner is a new role, which may well take responsibility as an intermediary, as a BPO operator or application provider to provide the entire life cycle of a service.
- **Collaborator -** the essential part of a service is that there are two parties to the use of a service. The collaborator is tracking a parallel view of the service throughout the life cycle.

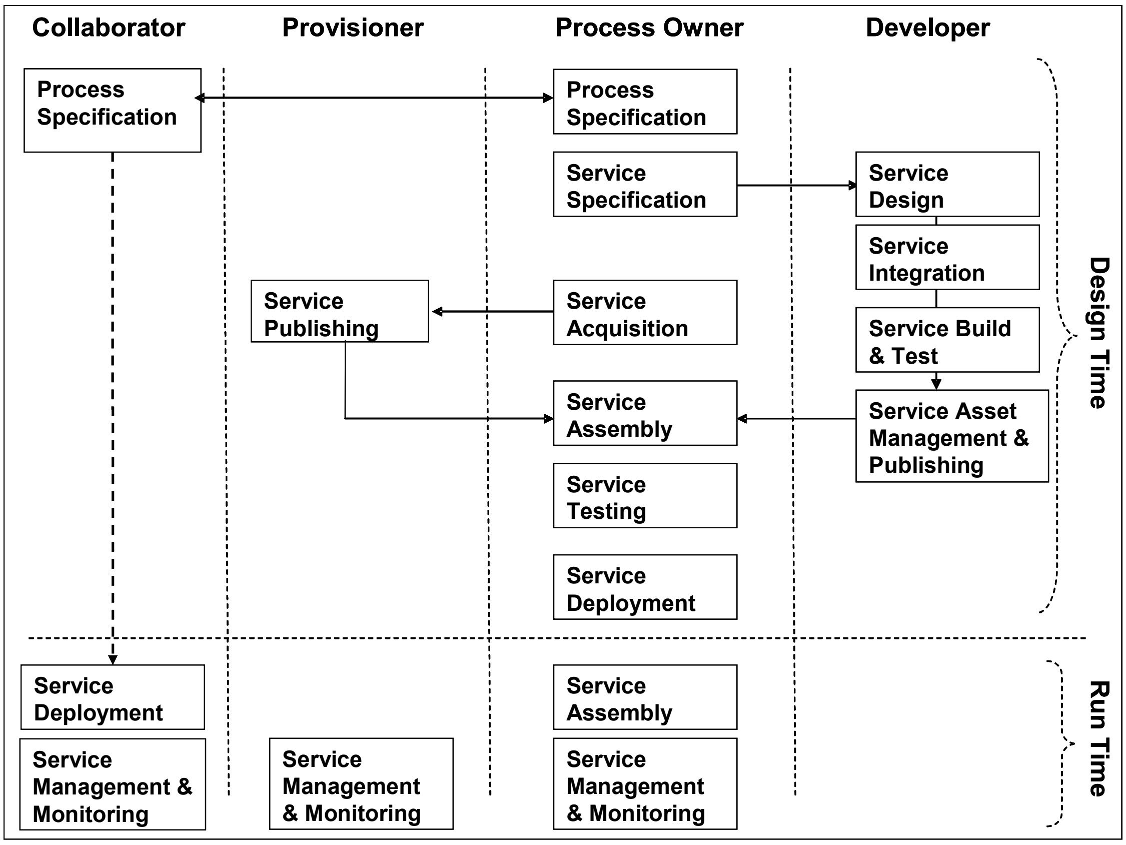


Figure 2 - The Evolving Organization

As the organization matures its use of services, roles and responsibilities will change. For example who undertakes the testing process? In the early stages as services are part of projects, budgets may dictate that the services are simply developed and tested and consumed as part of a common process – Though we might actually disagree and would advise against, but we accept reality sometimes makes this difficult. But as services become shared between projects, then across the organization and SLA's are required, it is essential that some form of (independent) certification comes into play.

Classification systems

Services are only useful if they can be found, and managed. Many are dismissing UDDI because they think that they will only use services that they already know about. They are ignoring the power of a directory as the basis of managing the service life cycle.

Our observation is that classification systems are going to evolve. It's important to set policies for specific issues such as versioning, usage, certification and so on, but it's also very important to allow developers and projects to extend the classification system to accommodate new requirements.

Actual QoS; Certification; Platform Class; Conformance; Configuration; Criticality; Deployment; Capability; Funding; Ownership; Pricing; Target QoS; Taxonomy; Security Class; Service Level; Usage Policy; Version

Box 1 - Example Classification Criteria

Service Level Management

SLM is closely related to classification. Part of a classification system needs to provide information on service level policies. In most enterprises the SLM policy will be established on an enterprise basis, covering two primary dimensions:

- Service trust - allows the provider to offer the service in a manner that is fit for purpose, and for consumer to easily understand the context in which it can reasonably be used. Service trust classification data includes - source, certification, protocol profile conformance, warranty, reusability, customer rating, security etc
- Service operations - allows the provider to establish standards, practices and platform products that are clustered around levels of operational guarantee. Service operational classification data includes - availability, reliability, performance, capacity, security, pre-requisites etc

Early learning and integration stage activity will use SLM classifications that have been developed specifically for the enterprise. However standards will evolve in this area, particularly in industry sectors and ecosystems where collaborations are implementing wide area business processes.

Monitoring and Recording

Services are a product, and the separation of provision and consumption creates the obligation on the provider to deliver service according to some form of agreement. Standardized classification taxonomies for operational guarantees and service trust characteristics provide the basis for enterprise standard monitoring and recording. A key part of establishing confidence in a Web Service is providing the infrastructure that allows customers and suppliers, and just as important disparate parts of the enterprise, with consistent and timely feedback on the performance of services provided.

Marketing

Yes, services are going to need marketing! Services are products, and must be sold to potential consumers. The prior two topics, classification and monitoring are the basics of marketing, - it works, let me prove it to you! However beyond the basics there is a requirement for a product life cycle approach to services. For some organizations this is as simple as managing specification in a collaborative manner that maximizes reuse. However for others the service will be a front line business product, where the technical aspects of the delivery are really trivial, compared with the complexities of market research and development. Our report on the BT authentication service[4] is a good example of this latter category, and is covered in the next topic.

Services as business products

Financial services organizations in particular have for some time now understood the concept of information as a product. Some organizations such as Amazon are moving rapidly to use the easier and deeper integration capabilities of Web Services to create extended products, particularly for their affiliates. It is notable that in the Amazon case, this productization is clearly early learning for Amazon also. However by the time organizations reach the third, reengineering stage of Web Service usage, it is to be expected that many enterprises will be offering Web Services as products, or as an integral part of pre-existing and or new core business products. What is particularly important to note here is that innovative enterprises such as Amazon, and BT mentioned earlier, have already embarked on R & D for integrated products. The lesson that financial services organizations have learnt long ago, is that the critical path is the business product, not the technology. Over and again we hear the comment that the technology is easy by comparison.

Security and Trust

In the past we have protected enterprise assets by erecting high walls which prevent access at a transport level; it is assumed that inside the high walls everyone is to be trusted. Today we

4 CBDI Journal Report - The BT Authentication Service, April 2003 http://www.cbdiforum.com/secure/interact/2003-04/bt.php3

recognize the inadequacies of these assumptions. We now understand that blunt instruments are inadequate, and that finer grained control over access and usage is essential. Web Services provide that finer grained control, BUT have the potential to increase exposure to greater levels of risk.

We also understand that security is never absolute; the best technical security will always be under threat from determined individuals. What we need are levels of protection that are relevant to an individual business service, which are an intrinsic part of the service design. The application level security oriented business logic, is then used in conjunction with broader grained security mechanisms which in combination create a trusted environment. Security creates barriers, trust establishes confidence that the service provides the advertised capability with understood and acceptable levels of risk.

As a consequence, delivering security and trust is increasingly the concern of everyone involved in the service life cycle. In planning Web Service and SOA related activity, it is a vital task to establish policies that define acceptable levels of risk in context with the specific business, and to implement these with new or altered roles and responsibilities as appropriate.

Infrastructure Stream

Web Services and SOA require new and modified infrastructure support. The Infrastructure stream provides a clustering of topics and deliverables required to establish these.

Provider Host environment

Hosting requirements will be more easily managed if there are clear policies set on service levels (see Planning & Management Stream). Service level classes may distinguish between broad classes of requirement, for example development, pre-production, low volume-low criticality production, mission critical production. However these could equally be application specific.

In the early stages of Web Service usage the primary requirement will be to provide a platform that makes it easy for developers to publish services. Requirements over time are almost certain to vary. A common approach to early learning stage hosting is to support a Web Service facade deployment with remote access to function and data. However as and when Web Services become integral to business critical functionality, with high volumes etc, it may become necessary to implement a management infrastructure which has greater visibility of the supporting functionality.

Longer term there will be a shift towards an 'On Demand' operating environment, where not only will the physical location of the service implementation become increasingly remote from the business service provider, but will evolve to become dynamically locatable.

Consumer environment

Though we expect the consumer environment to be a mirror of the provider's in terms of the logical capability, we make the distinction between them because in some situations such as small mobile devices the requirement for purely consumption will drive a different set of service characteristics

Middleware

Many organizations will already have implemented middleware backbones or a bus structure that allow intra and inter enterprise communications. Moving to Web Services and SOA requires new protocol support and a number of additional services.

Integration and Assembly

In the early learning stage Web Services are generally an additional option for integration. Many organizations may set the objective to establish an assembly infrastructure and environment that allows services to be used in many different contexts. It should be noted however, that many of the early services may not be good candidates for general purpose assembly, unless they have been reengineered for general purpose usage.

Topic Area	Deliverables	WS Maturity Phase			
		EL	INT	RE	MAT
Provider Host environment	WS Developer environment	Y			
	Hosting environment for WS facades		Y		
	Hosting environment for WS applications	Y	Y		
	On Demand Operating Environment	Y	Y	Y	
Consumer environment	Internal	Y	Y		
	External				
Middleware	Middleware and Communications technology	Y	Y		
	Broking and Routing				
	Transformation				
Integration and Assembly	EAI use of Web Services	Y	Y	Y	
	Workflow and BP standards		Y	Y	
	Aggregation and composition mechanism				
Development environment	Tools policies		Y	Y	
Asset Management	Asset management environment supporting supplier/consumer environment	Y	Y		
UDDI Directory	Static usage support	Y	Y		
	Dynamic usage		Y	Y	
	Publishing process				
	- manual process				
	- more automation, integrated with Asset Mgt.				
Service Level Management	WS Management environment		Y		
	SLA management		Y		
	- Provided				
	- Consumed				
Security Infrastructure	- XML Firewall		Y	Y	
	- Message level security			Y	
Monitoring & Measurement	Mechanisms to provide		Y	Y	
	- SLA compliance				
	- Service level logging				
	- Business monitoring services				
Diagnostics, failover	Fault management		Y		
	Diagnostics and Alerts				
	- Infrastructure				
	- Business Service				
Consumer/ Subscriber Management	Access control		Y	Y	
	Subscription				
	Accounting				
	Common billing system				
WS Protocols	Standards adoption and compliance		Y	Y	

Table 3 - The CBDI Roadmap Framework – Infrastructure Stream

Development environment

Development tools have rapidly adapted to provide support for Web Service protocols. Through a combination of platform and development tool support, the technology of Web Services will be largely transparent to developers. Adequate support is already provided for the early learning stage. However, beyond that tools will need to evolve further to provide support for SOA modeling and design, and the ability to support the design of complex service based business processes that must be translated into emerging Web Service protocols such as BPEL.

Moving further out, we can expect to see much greater emphasis and support on the assembly and consumption of services integral to business process life cycle support tools. As the coverage and sophistication of this support extends, we will view this as a service assembly and management platform.

Asset Management

As Services become further abstracted away from their implementation through the application of SOA principles, they will be recognized as an independent asset type in their own right, rather than a property of a component or object. Asset Management will become important during the Integration stage to understand and manage the complex relationships between requirements, services, implementations, and countless associated development artifacts.

UDDI Directory

The emergence of private UDDI directories in enterprises will become widespread during the Integration stage to facilitate the internal provision and consumption of Web Services, and to provide a filter to external unqualified Web Services in public directories.

Service Level Management

Service Level Management will be essential for both Service Providers and Consumers to monitor Services and their compliance with Service Level Agreements. Whilst the growing volume of Services in use and transactions will be obvious drivers to SLM, even the simplest of Web Services used during the Early Learning stage may still be subject to stringent SLA requirements and need careful monitoring.

Most enterprises will implement Web Services Management capability through a variety of approaches to enable the following topics including security, monitoring and measurement, diagnostics etc.

Security Infrastructure

Security is often referenced as a barrier to wider Web Services adoption. In the early learning stage existing web security mechanisms such as SSL will often suffice for straightforward transactions. WS-Security protocols will enable message level security that is suitable for more complex transaction scenarios. It can be expected that malicious attacks, such as denial of service will start to use Web Services necessitating the use of XML aware Firewalls.

Monitoring & Measurement

Mechanisms will be needed quite early in the process to monitor Web Services for SLA compliance, and straightforward fault management and diagnostics.

Where Web Services are abstracted away from the underlying low level network and database traffic that composes the implementation, it provides an opportunity to monitor Services at a meaningful level to the business. This enables not only monitoring of computing resource, but also of business performance.

Diagnostics, failover

With meaningful business Web Services, diagnostics and failover can be considered at both the business and infrastructure levels. Failover can be implemented at the SLM level for example by re-routing Service requests dynamically.

Consumer/Subscriber Management

The need to manage service consumers in terms of access control (not the same as security) and accounting for their usage is not just relevant to commercial Web Services, as internal usage may still be the subject of internal accounting procedures. Not only may this entail providing links to accounting and billing systems, but to provide a more dynamic environment, self subscription mechanisms will also have to be provided to consumers.

Web Service Protocols

Though several core WS protocols are in place and/or in progress through standard committees, a broader set will continue to emerge and evolve to support the needs of Enterprise level Web Service and other scenarios. Many organizations are implementing their own protocol compliance guidelines in order to facilitate early learning activity. It will make sense to adopt the WS-I profiles as they come available, although it will still be necessary to have some level of coordination on what profiles are being used at any point in time, and also any local overlays that are applicable. Interoperability will continue to be a minor problem in the Early Learning stage.

Architecture Stream

Topic Area	Deliverables	WS Maturity Phase			
		EL	INT	RE	MAT
Service Oriented Architecture	Patterns, design guidelines Life cycle management guidelines including versioning	Y	Y	Y	
Trust & Security Architecture	Security framework and common services Trust design patterns	Y	Y	Y	
Business Service Bus	Defined service sets		Y	Y	Y
Redefining the Application	Project scoping guidelines		Y	Y	
Governance framework	Guidelines, patterns and practices to ensure implementation of SOA policies		Y	Y	
Shared usage	Common Services at infrastructure and business levels	Y	Y	Y	
Semantics	Policies governing semantics Defined semantic sets Collaboration projects - industry; organization; supply chain etc		Y	Y	Y
Protocol coordination	Managing compliance Best practices to ensure interoperability Defined profiles and transition coordination		Y	Y	Y
Information Ownership	Information Currency Strategy Review of Information ownership Common Information access services - rationalize or wrap existing sources		Y	Y	
WS Based BI Strategy	BI Patterns BI Priorities	Y	Y	Y	

Table 4 - The CBDI Roadmap Framework – Architecture Stream

In the early learning stage, Web Services will typically be adopted in a tactical manner. Architecture is not a significant issue. As an organization moves forward it is looking to integrate

Web Services into core business processes, to progressively establish the service as the unit of reuse across an organization. The Architecture stream provides a clustering of topics and deliverables to support this activity.

Service Oriented Architecture

SOA is the broad set of concepts that enable units of functionality to be provided and consumed as Services. This essentially simple concept can and should be used, not just for Web Services, but also at each tier of the architecture, in order to compartmentalize and provide flexibility.

Establishing a Service Oriented Architecture is a foundational activity that forms the basis for a more adaptable application portfolio, exposing core functionality as Web Services. CBDI advise that creating an SOA is a medium term project, which implements services to a common bus structure (policy, semantics, middleware). This is typically an integration stage Roadmap activity, and should be viewed as an essential pre-requisite for the reengineering Roadmap stage. More details can be found in the section "Moving to SOA"

Trust & Security Architecture

Application publishing and using Web Services require a new layer of security that is separate from the network firewalls, which in the main can do no more than block unwanted protocols and rogue IP addresses. There is an interesting conflict that in order to empower an application, the credentials and encryption capabilities have to be moved nearer to the code and away from the infrastructure. But, to maintain a clean separation of concerns the service implementation must be clearly separated from the security management layer. The new SOAP protocols for WS-security allow an application to deal with data that is private, right from the point of entry, all the way through to the point of delivery, and even then it can remain encrypted in storage. Similarly, authentication is end-to-end, from the individual that signs the request right through to the business process that checks the ID. This is how you do conventional business – you sign the check, not the postman.

In the Plan & Manage Roadmap Stream we identify the business and technical policies that drive the application specific security. Here in the Architecture Stream there is a requirement for clear architectural guidance in various forms, to ensure conformance with the policies and clean separation of concerns.

Business Service Bus

In the days when middleware was top of the technical architect's toy box, the notion of the transaction bus was very popular. But whilst this is a necessary layer, it's equally if not more important to develop the concept of the Business Service Bus. The Business Service Bus is the set of business services for a specific domain that are available for widespread use across an enterprise. The services are published in a UDDI compliant registry which allows them to be reused without manual intervention by the provider.

The creation and particularly the population of the BSB is an important step towards service maturity, which facilitates widespread reuse of common services.

Redefining the Application

Application scoping has always been an imprecise science for end user enterprises and ISV's. In the end result many projects are scoped around a compromise of functionality and commercial/budgetary considerations that makes most sense at the time.

In the initial stages of service usage, the service is simply an appendage to existing applications. However the service concept is closely related to the business process; some say the service is a business process. Over time therefore sets of service will become the natural focus of application investment and scoping. Why over time? Because there are relatively few applications being built from scratch today, and it will take time for services to be incorporated into existing portfolios, to a point where there are service based artifacts available as a basis for planning.

Governance framework

An important aspect of SOA is the question of how to ensure architectural decisions get implemented - otherwise known as the governance issue. Each architectural deliverable can be attributed with governance roles, for example - standard or guideline, mandatory or optional. These roles then have applicability to specific domains which might include platform, product, layer, application, relationship etc. Governance may then be managed by a variety of mechanisms which include patterns, templates, common components, common services, protocols, semantics, products and practices.

Shared usage

Creating services for use by multiple consumers has all the complexities of creating common components, and then some more. This is a major area of architectural consideration, which will need to be planned for each application and service set.

The first thing to think about is generalization of functionality. This is one of the primary issues in the integration Roadmap stage, because the level of generalization is determined to a large extent by the existing applications. Some level of generalization and transformation can be built into the facade and business process scripting layers. However this may not be possible without expensive, invasive modifications. So in the early stages there may be considerable compromise required.

CBDI defined the concept of the differentiated service[5] as an important pattern for service architects. The differentiated service is a device for managing reuse of common services in a context sensitive manner. This pattern may be used to simply reduce the number of interfaces, and to increase the applicability of an individual service offering. More importantly the pattern will be applicable to design guidelines for security and productized business services.

Semantics

Web Services enable wider involvement in existing processes and or wider use of standard services. These developments require that the users have a common language system. In simple domains, it may at least be theoretically possible (even if often impractical or technically inappropriate) to establish a single shared (global) data model, so that all local models are mapped to the shared model. In complex domains, which means most large enterprises, this isn't even theoretically possible. There is no top-down approach, because there is no single agreed place that is the "top" - merely lots of different stakeholders each of whom thinks the world revolves around his agenda.

Many organizations have commenced some rationalization at a document level as they implement XML based transfers between systems. However many of these implementations have been achieved using transformations rather than rationalization. A key architectural task for organizations preparing for widespread usage of Web Services is to establish the policies and practices that lead to rationalization where it makes business sense, and for coordination of different stakeholders views.

In the CBDI Report on Data Collaboration[6], we examined how a distributed business process can be understood as a series of collaborations or conversations between different stakeholders. Each stakeholder has a context, which can be expressed as a local data model. This leads to defining firstly the documents that represent the exchange of information and services between stakeholders, and secondly the process wiring that bridges between the multiple contexts. We look at the techniques for doing this, and the implications for decomposing requirements into discrete services?

Protocol coordination

Whilst it is probable that there will be universal consensus around the transport protocols, SOAP and WSDL, it is by no means certain that this acceptance will apply throughout the entire Web Service stack. Although ebXML has adopted the transport protocols, it is entirely feasible that it

5 Design Pattern: Differentiated Service, CBDI Journal December 2000
http://www.cbdiforum.com/secure/interact/2000-12/design_pattern.php3
6 CBDI Report - Service Identification - Data Collaboration, March 2002
http://www.cbdiforum.com/secure/interact/2002-03/service.php3

will continue to occupy a niche, particularly in business contract descriptions. Likewise there is no evidence yet of agreement around BPEL, WSCI etc.

Each organization needs to select the combination of protocols that are likely to be most applicable in terms of functionality and acceptance within the ecosystem relevant for that enterprise.

Information Ownership

Replicated data is the industry convention. We replicate because there has been no sensible alternative. Yet replication is an inherently weak and error prone model that causes inconsistency, complexity and huge costs. Web Services based interoperability offers interesting solutions to this problem as the new SOAP based Internet standards enable pervasive accessibility. We don't suggest that all data replication will or should be eliminated, but we do anticipate opportunities to simplify and improve business processes that architects should be looking for right now.

Over time data distribution will migrate towards the point of ownership, or a proxy of same. Rather than every organization replicating and maintaining copies of the data, real owners of the data will maintain it themselves and make it accessible to everyone who needs it. Your bank does not 'own' your name and address, nor does the retailer, but you do as an individual. They do own the transactions you make of course, but this is an intersection between their products and you. What if instead, you maintain your personal information in one location and everyone retrieves it from there at the time they need it? All any interested parties have to do is store the URL that you give them and then retrieve up to date information whenever they need it.

Not every piece of information needs to accessible in real time. Where it is appropriate the migration to a Web Service will still be gradual. But this is a progressive process which will be driven by business opportunity and need. An important part of Roadmap planning is therefore to prepare for the progressive change, a) by providing the infrastructure to allow data migration and b) working with business units, partners, customers and suppliers etc to improve the currency and accuracy of data using Web Service based architectures.

There is comprehensive treatment of this topic in the CBDI Report “Will Web Services Revolutionize Data Distribution?”[7]

WS Based BI Strategy

Web services are being widely discussed for integration of operational business processes. Many enterprises are starting to deploy Web Service technology for connecting applications internally. Whilst there is significant interest in deploying the same technologies externally, this is currently inhibited by concerns such as security and reliability. Meanwhile, the use of the Internet as a platform for business intelligence (BI) is becoming more mature and sophisticated. There is an important role for Web Services in the business intelligence space based on the capability of creating real time, rules driven information access services, which can potentially deliver real business benefit even with relatively unsophisticated WS technologies.

At the core of most BI systems are software products providing query, reporting and analysis functionality, sometimes referred to as OLAP. Traditionally, these functions have been based either directly on operational systems, or more commonly on a data warehouse or data mart, which assembles and restructures data from one or more operational data stores. The result of the data combination can be stored in another data store, or may remain virtual. However business intelligence should be viewed as a closed control loop system where managers use tools to process and interpret information; they then act upon this information and monitor the effects of their actions. If the actions have the expected effect on business performance, this helps to confirm the original interpretation; if management intervention doesn’t work in the expected way, then this should trigger further analysis. This management feedback and learning loop is a key element of true business intelligence.

7 CBDI Report - Will Web Services Revolutionize Data Distribution? December 2000
http://www.cbdiforum.com/secure/interact/2000-12/rev_data_dist.php3

It is also important to provide a context for making sense of events and trends. Regular readers of CBDI reports will recall our analysis of the Kodak case[8], in which the online retailer found itself obliged to supply a large number of digital cameras at an incorrect price. Real-time business intelligence can help identify a sudden increase in demand for a given product, and may place some constraints on automatic supply until the increase can be explained. But we need some context for this. It is only when we can link a sudden increase EITHER to a marketing campaign by Kodak OR to some hostile activity on an internet newsgroup that we know what to make of it - and therefore how to respond. For more information on this topic there is fuller treatment in a related CBDI report[9].

Process Stream

Topic Area	Deliverables	WS Maturity Phase			
		EL	INT	RE	MAT
Basic Life Cycle definition	Communication baseline		Y	Y	Y
Business Integration (Services as business products)	Product line (or equivalent) process			Y	Y
Collaborative (Supplier /Consumer) Life Cycle	Documented Life Cycle with applicability guidelines		Y	Y	
Collaborative specification	Specification guidelines		Y	Y	
Certification	Certification requirements and relationship to SLA levels - architecture review - business logic - performance - security and trust		Y	Y	
Publishing	Publishing Guidelines and repository requirements Life cycle management process		Y	Y	
Deployment and Versioning	Deployment practice and guidelines	Y	Y		
Acquisition	Agreement templates		Y	Y	
Security and trust	Security and trust process guidelines		Y	Y	Y

Table 5 - The CBDI Roadmap Framework – Process Stream

8 CBDI COMMENTARY - WHERE DID THEY GO, I JUST DON'T KNOW . . . February 2002
http://www.cbdiforum.com/public/news/index.php3?id=885&news_date=2002-2
9 CBDI Report - Web Services To Improve Business Intelligence, June 2003
http://www.cbdiforum.com/secure/interact/2003-06/bi.php3

In the early learning stage, Services will be managed as adjuncts to conventional systems delivery processes (such as process design, application development, configuration and deployment etc). However, as Services become the primary mechanism for inter company communication, it will be essential for some considerable process reengineering to take advantage of the service orientation. A fundamental premise of the service is also that there is a provider and consumer, each of whom is executing different processes; the challenge is to align collaborator processes. However in the Roadmap process it is important to differentiate between use of Services as technical interoperability devices, and use as black box components where provider and consumer processes are quite separate.

Basic Life Cycle definition

A basic life cycle is required to cover use of services in project and integration use. The scope of the basic life cycle is likely to include the following deliverables:

- Definition of generic Service delivery process
- Operational Service contract management guidelines
- Security patterns and guidance on application
- Service integration patterns and guidance on application
- Service monitoring best practices and guidelines
- Exception handling best practices
- Diagnostics best practices
- Quality assurance and testing best practices
- Change and version management best practices and guidelines

Business Integration (Services as business products)

SOA today is a largely technical concept surrounding the creation of loosely coupled services. Further SOA is often regarded merely as a superior way of deploying a component-based architecture - and in their early use of SOA, many organizations will not go far beyond CBD. However the real benefits of services will emerge when enterprises understand the concomitant business transformation that will be enabled.

In the old economy, a standard enterprise transforms raw materials and components into finished product. In its simplest form, the business process can be represented as the kind of production line designed by Henry Ford. In the service-based economy, the core business model is no longer based on the conversion of raw materials into finished goods. The enterprise buys in services and sells services, and the core value proposition is the conversion of input services into output services. This calls for a very different kind of business process model, which allows us to see how services combine and interact to make other services. For example:

- A service may be subject to continual maintenance by the service provider, constantly striving for improvements in scope and performance, so that the same service can be offered to an ever larger number of consumers.
- And the consumers of a service have much greater flexibility to switch between alternative services.
- The selection of a service may be done in real time, based on some set of policies.
- If a service is to operate differently for employees and for other users, there has to be some context model that shows how employees are to be differentiated from other users.
- A service should be designed to operate in as many different contexts as possible. This requires some modeling of the service context. It may also require some modeling of the differentiation policies to be applied when executing the service.
- The importance of the contract in SOA cannot be overestimated. The formal contract is the device that allows us to create virtual businesses; formalize system scope and boundaries; minimize dependencies and thus maximize adaptability; use blackbox testing and have choice of Services and easier change of supply.

In considering the Web Services Roadmap, it will be necessary to consider how business services may alter the core and peripheral business of the enterprise(s) in question, and crucially in what timeframe.

Today we model requirements using artifacts that are appropriate for understanding how to create applications and components. As our use of services matures, we will need to adapt our modeling techniques appropriately. For example hiding complexity (which is what a service does) yields a gain in manageability, as well as flexibility and reuse. Business analysts should consider carefully which type of complexity is most at issue in a given situation, and choose the appropriate modeling approach accordingly. See related CBDI Reports providing insight on this topic[10].

Collaborative (Supplier /Consumer) Life Cycle

In the early learning and possibly the integration stages processes used to deliver, manage and consume Web and other types of services will be progressive extensions of existing processes and practices. However the real benefits from Web Services will only be realized when the providing and consuming processes are reengineered such that the interdependency between provider and consumer is reduced to the minimum necessary.

Figure 3 illustrates that we have parallel universes that are focused on different aspects of the same service. Once an organization has implemented this perspective thoroughly, their ability to use, reuse, outsource, change supplier of services is radically improved.

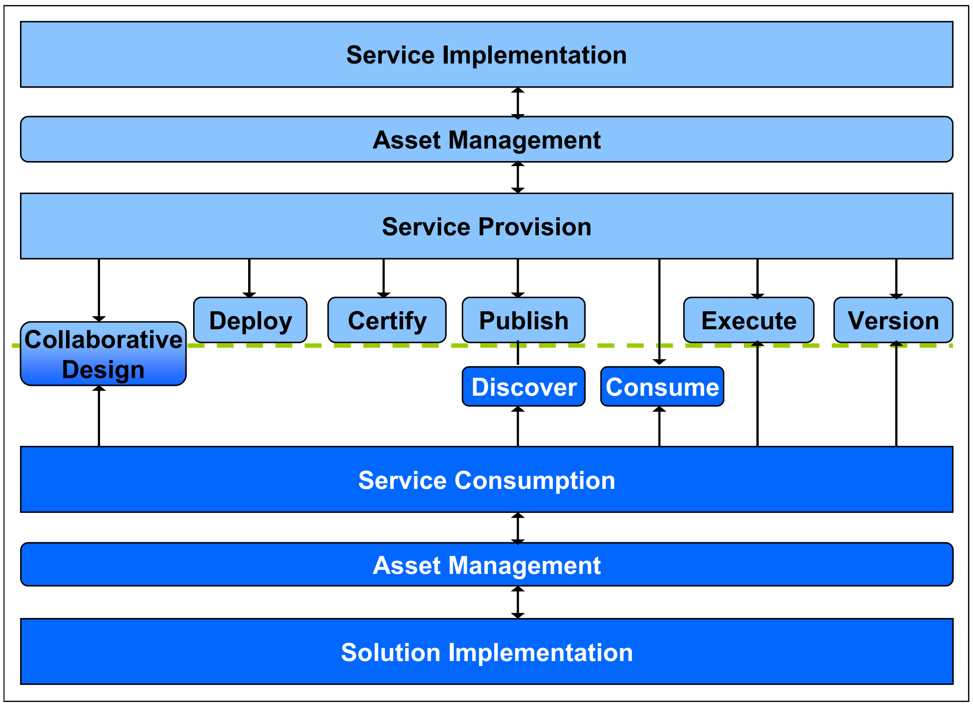


Figure 3 - The Service Life Cycle

Organizations should be responding to a number of drivers as follows:

- As Services become more central to business and IT architectures, an holistic process that manages the Service life cycle will become essential. Management of the specification, delivery, acquisition and consumption of the components should be coordinated by a Service life cycle perspective.

10 CBDI Report - Modeling for SOA, February 2003 and CBDI Report - Modeling for SOA - Worked Example, April 2003 http://www.cbdiforum.com/secure/interact/2003-04/model_soa.php3

- Services will increasingly be provided and consumed on an inter business basis. To facilitate this, certain aspects of the Service delivery processes implemented in each of the participating organizations will need to be common in order to ensure reasonable productivity and to deliver the required levels of trust and security.
- Services will increasingly be designed through a process of collaboration between multiple participants, with increasing emphasis on vertical industry standardization of business Services and Service based business processes. A mutually understood Service delivery process is essential to ensure successful collaboration
- With increased collaboration between fundamentally separate business entities, there will need to be greater formality in the specification of services, contracts, etc. Achieving this will drive the requirement for some formalization of the Service Delivery Process and mutual understanding of deliverables
- With Services provided to external organizations, sometimes on a commercial basis, far greater emphasis will be placed on risk management, quality of service, service level agreements and accounting. The Service delivery process must encompass this across the entire lifecycle, for example as part of the Service strategy and design, not just as a facet of operations.

The scope of the collaborative life cycle is likely to include the following deliverables:

- Definition of generic Service delivery process
- Definition of generic commercial process
- Identification of deliverables
- Clarification of organizational roles such as
    - provider
    - intermediary
    - consumer/requestor
- Clarification of responsibilities
    - Service specification
    - Contract architect
    - Service publisher
    - Service manager
- Definition of publishing best practices
- Business Service contract management guidelines
- Operational Service contract management guidelines
- Trust patterns and guidance on application
- Security patterns and guidance on application
- Service integration patterns and guidance on application
- Service monitoring best practices and guidelines
- Exception handling best practices
- Diagnostics best practices
- Quality assurance and testing best practices
- Guidance on making semantic agreements
- Service level agreement best practices and guidelines
- Change and version management best practices and guidelines
- Business process design patterns for inter company exchange

Collaborative specification

If businesses are to collaborate using Web Services to support a business process, this raises the question - how is this going to be managed. Obviously we don't want to hard-code Web Service calls into an application, because this will be highly inflexible. Instead, we want something like a scripting language, which will define the wiring between Web Services in a way that can be interpreted and executed at runtime.

BPEL4WS, the Business Process Execution Language for Web Services is just that - a workflow definition language that allows businesses to describe complex business processes capable of both consuming and providing Web Services. BPEL4WS represents a merger between two initiatives: IBM's Web Services Flow Language and Microsoft's XLANG. The BPEL4WS scenario can be understood as a programming abstraction of a long-running business transaction or collaborative business process, and is sometimes known as choreography.

For distributed collaborations, a contract between provider and consumer needs to have three parts. The first part of the contract specifies the function of the Service, typically using a series of logical assertions such as preconditions and postconditions. The second part of the contract specifies the quality of service - which are often given the rather dismissive label of non-functional characteristics. The third part of the contract specifies the commercial arrangements, including charging for normal operation and compensation for abnormal operation.

At the time of writing the BP scripting languages have very limited support for contractual specification. However in time this will be an area of critical focus, because it will be the mechanism by which we articulate precise obligations between collaborators. This collaboration will be an area for significant process development and improvement in the reengineering stage of the Roadmap. CBDI have undertaken research in this area, and organizations considering process patterns in this area may consult the related CBDI Report[11].

Certification

Service Testing and certification practices require careful examination. Whilst many other issues relating to Web Services may be safely delayed until volume and widespread usage force action, testing and certification need early attention in order to ensure that Web Services are widely used and reused. The big issue is will the service work correctly every time when I need it?

A further aspect is that Web Services are by definition designed in isolation from the end user. What we are asking for is the same level of reliability from "ordinary" business components as you would expect from safety-critical systems. We would all like 100% correct components, but what happens when the environment in which that component is being used changes? A component that is 100% correct for internal use can become vulnerable and a weakness when used on the open Internet, particularly when trust levels change. Even 100% correct can be wrong if the specification is wrong. This is a major problem with software in general - how do you test the specification? The Ariane 5 software was 100% correct to its spec for Ariane 4, but was reused in a different environment and caused a 500 million dollar uninsurable rocket launch to explode because of an integer overflow problem. Both computers worked perfectly, both shutdown correctly, and the self-destruct mechanism triggered.

Web Services therefore place new demands on testing activity. There are fundamental differences between black box and white box reuse, and whilst we have all paid lip service to black box with components, once we go beyond the boundaries of project reuse, we are forced to operate with full transparency between provider and consumer in the Services world. Testing seems too narrow an expression because traditionally it is limited to activities that take place in the development process, and not during the execution phase, and certainly not prior to every business event. We need some form of active certification that the task has completed correctly with no unwanted effects. This goes beyond simply checking return codes and provides confirmation that the business transaction has completed correctly, which may of course involve multiple parties to validate this.

---

11 CBDI Report - From Web Services to Web Collaborations, November 2002 http://www.cbdiforum.com/secure/interact/2002-11/collaboration.php3

The need for dynamic testing suggests we need to look at each part of the life cycle of a service and identify appropriate testing and certification at each stage. This requires a new framework for testing which might include:

- Static Testing - execution of a service prior to deployment
- Static Certification - warranty by some certifying body that a service is fit for purpose prior to deployment
- Dynamic testing - execution of a dummy or test service, that is used as part of pre-conditional activity to ensure the service provides the correct functionality.
- Dynamic certification - execution of post conditional services that verify the service has executed to specification.

The dynamic aspects of testing XML Web Services require the introduction of testing services in addition to the functional service. These are pre-condition and post-condition checks before and after each functional interface, and could include technical and business oriented testing services.

A technical testing service may check that the WSDL complies with prior agreements. A business testing service would ensure transactional integrity. For example if a bank provides a web service function call “fundTransfer(fromAcct, toAcct)", then there may be two additional services required:

> "checkAccts(fromAcct)" which checks the fromAcct has enough funds, and "checkBalance(fromAcct, toAcct)" which verifies the sum of the two accounts is unchanged.

These additional services are specifically checking pre and post conditional states, and designed to validate the correct operation of the core business service. A benefit of separate classification of these services would be to make it easy to manage (the entire life cycle of) these services quite separately from the core business transactions.

Publishing

Web Services that are intended to be consumer by others need publishing. The extent and quality of the information will reflect the level of separation between provider and consumer, and also the SLA underlying the service.

In the early stages of Web Service use, many enterprises find they do not require discovery facilities provided by UDDI, because services are communicated to potential consumers that are already known.

However as soon as the service is offered on a broader basis the publishing process needs to expose sufficient information to allow the user to understand the capability being offered and the implications of usage. Published information may also vary depending on the product SLA offered, for example more comprehensive information on the service behavior as a UML model, which may be appropriate to certain classes of consumers.

Deployment and Versioning

Web Services protocols have no explicit support for versioning. This is not a particular issue while Web Services are being used internal to projects, where provider and consumer are one and the same. But as soon as the provider and consumer are separate entities, and particularly when there is automated consumption, that is no communication other than provided by the WS protocols, then a versioning strategy is essential to ensure that there is good version coordination.

It's important not to confuse the version of the implementation with the version of service (interface). New versions of the implementation should be transparent to the consumer. The service interface can of course be extended without impacting existing consumers, and it is common for projects to consider versioning a non issue because of that. But the issue is that you still have to ensure that consumers are made aware of extended functionality. Further there is a question of traceability and supportability. How do you know who's using what and why, and from what point in time? So whilst Web Services, courtesy of XML, are loosely coupled and won't break contracts so easily as tightly coupled services, nether less you still need a change management process to trace changes, notify service consumers where necessary and track usage.

Acquisition

The technology issues involved in acquiring Web Services are pretty easy. If you are consuming a Web Service on a sold as seen basis, such as an exchange rate calculator, or a weather forecast service, and you are quite happy if it may not be available in the future, or perform in exactly the same manner, then acquisition is easy.

Widespread acquisition of services is unlikely to become widespread business practice in the near future however, not because of security, but because of external dependence and the risks associated with placing business reliance upon third parties. But not all services have the same dependence characteristics or profile, and in the table below we provide some thoughts on where you might want to go faster or slower in your thinking.

Class of Acquisition	Concerns	Stage
Commodity service acquired singly or in narrow footprint, from major supplier. For example authentication service or Amazon's Associate service	High dependence requires confidence in service level. However contractual agreement probably determined largely by supplier, who will rely on reputation rather than contractual agreements. Alternative sourcing is viable back-up and cost management action	Major suppliers will be in a good position to provide trusted services on the basis of mass adoption. Serious options will apply in 2nd Stage
Sets of commodity services acquired from single source. Example Salesforce.com CRM and sales management services	As above	Whilst vendors such as Salesforce.com are making progress in this area, particularly in specific horizontal applications, widespread adoption will be slower
Sets of custom or limited volume services acquired from single source. Example vertical specialist services	SLA applicable to the entire set reduces administrative overhead. Commitment to one source makes SLA high criticality.	Stage 3
Ad hoc single service acquired on demand. Example actual calculations, weather forecasts for specialist purposes	Behavioral more important than operational guarantees. Multiple sourcing options based on differential behavior	Stage 2

Table 6 - Service Acquisition

Security and trust

Security is a core topic in Web Services land. We have discussed security implications in planning & management, infrastructure and architecture. The purpose of a Web Services security architecture is to enable a variety of Web Service implementations to securely interoperate in a platform - and language-neutral manner, and to ensure the integrity, confidentiality and security of Web services.

The real issue at the heart of these cases is that our systems automate standard or expected behavior. We omit to implement controls that "monitor" the behavior of systems. We hope that nuclear power stations have these controls, but we don't apply the same logic to business systems. We trap software errors not systems errors. Web services will potentially magnify this problem. With Web Services we will be able to increase the automation of business processes within and across companies. The lesson for Web Service providers is therefore clear - they may be introduced to improve operational efficiency, but must also be complemented with intelligent monitoring and systems controls.

The purpose of process activity therefore is to ensure that the architecture is used appropriately. We might expect that the process includes steps such as:

- Define a set of business security requirements.
- Use the Web Services security architecture to design (some elements of) a security solution - at least for the web service elements of a business system.
- Verify and monitor adherence to the business security requirements, and detect any breaches.
- Diagnose breaches - how did this intruder get in, how did this information leak out - and plan appropriate corrective/preventive action.

Projects

Topic Area	Deliverables	WS Maturity Phase			
		EL	INT	RE	MAT
Experimental and early learning	Shared experience Services as a better form of project architecture	Y			
Project level services	Services as a better form of project architecture Services shared within project	Y	Y	Y	Y
Implementation Based Services	Harvesting Existing implementations/interfaces rendered as Web Services		Y		
Enterprise level services	Generalized services providing single source of functionality SLA based guarantees Formally published Managed upgrade		Y	Y	Y
Services are products	Integrated product and IT development process			Y	Y

Table 7 - The CBDI Roadmap Framework – Projects Steam

Many organizations will maintain recommended project profiles for project teams to then customize as appropriate. Existing project profiles will require customization and there are five important profiles that may be important to consider and plan for.

Experimental and early learning

Exploratory activity that is undertaken either by individuals of individual project teams. No significant reliance on shared infrastructure.

Project level services

Web Services created as interoperability mechanisms within the confines of a project. Relatively minimal requirement for formality of process, for example separation of supplier and consumer, certification etc

Implementation Based Services

Web Services used as integration layer for existing and legacy applications.

Enterprise level services

Web Services created for reuse across the enterprise.

Services are products

Web Services form an integral part of one or more core business product offerings.

A Web Services Maturity Model

A Strategic Perspective for Technology and Business Planning

Adopting Web Services and SOA is an evolutionary process for vendors and their customers. The question is how do you manage progress and risk in a constantly changing environment? While the technology issues are inevitably dominant today, this is merely a symptom of the immaturity of the service environment. Very soon business issues will dominate. In this report we introduce a high level maturity model that provides an assessment of relevant timescales and a framework for aligning business and technology roadmaps.

Introduction

Every parent is familiar with the sound of the tiny voice piping up from the rear seats of the automobile "when are we going to be there?" Many are starting to ask the same question of Web Services, which are on the face of it, taking an inordinate amount of time to come to anything like widespread acceptance. Believe it or not it's nearly three years since Web Services first came on the scene, and four since XML-RPC activity signaled the paradigm shift. Two things are clear - first Web Services are far from mature by any measure, and second we have a long way to run before we reach anything like maturity.

You might well ask, "What defines a mature state?" Also "What happens after we reach maturity?" Although, as our recent survey confirmed, many organizations are adopting Web Services, equally many organizations are deliberately adopting conservative, mainstream adoption policies, and currently staying out of the Web Services market. In a difficult economic climate, why take a risk if you don't have to?

Yet Web Services are a little different to many other new technology trends. Web Services are clearly evolutionary insofar as they enable wrapping of existing application functionality, and provide early ROI based on improved reuse and better structured applications. And with the standards process very obviously following a course of layering complexity upon complexity, it is clear that a prediction of concept maturity based on the standards would be 2005 at the earliest.

Is there a risk that Web Services will fail, and be superseded by something else or simply swept aside by some new technology trend? Frankly this is about as likely as a collision with a stray interplanetary body. With the hegemony of Microsoft and IBM driving Web Services at the core of their business strategy, this is a technology that is going to run and run. So the questions to ask are when do you jump on the train and why?

And here we need to consider the relative maturity of both technology and business.

Technology Maturity

If we look at Web Services as a set of technologies, then we should expect to see an Atlantic breaker pattern, where rolling waves are continuously breaking on the shore, with something resembling predictability and regularity. Because the evolution of Web Services is fundamentally driven by the standards process, we can forecast when the waves of security, reliability, management etc will happen. Although there are competitive and commercial games being played with the standards process, it is to everyone's advantage that there is universal buy-in to the WS protocols, and therefore we can be less concerned about tactical battles, because the outcome is pretty much a given.

With most new technologies there is an observable model where early use of any new idea is perceived as an extension to existing practices. So the first automobiles were designed to look like carriages without the horse. Mobile phones provide(d) text messaging facilities that use the numeric keyboard irrespective of the difficulty of using the UI.

Although the use of Web Services for better integration is now becoming widespread, the reality is they are simply another layer on top of the infrastructure that already exists. This is particularly true in the Java environment, where there are so many layer mappings - from UML, to relational, from XML to objects and back again, and XML to code, that it's not surprising that organizations are going slow, unless there is a real imperative. In an upcoming report we look in detail at new XML based implementation technologies such as X# and Water, which clearly have the potential to make large swathes of the current development complexity redundant. So while the technology world waits for the WS standards to catch up with expectations, we might just see a major advance in implementation technologies happening concurrently, which will deliver massive simplicity into the services "delivery" environment.

Business Maturity

In Richard Veryard's recent CBDI reports on Modeling for SOA[12] he has described the opportunities that are going to surface as we achieve the reality of trusted, and ubiquitous service interoperability. In particular he has emphasized the importance of looking at mundane business processes and services in a new context, for example looking at the service ecosystem as the scope for business design. What we used to call reengineering before it became over hyped and unfashionable!

An equally interesting perspective to examine is the way that business may be transformed by service thinking. One of the more interesting reports that I have personally researched over the past six months was about BT's authentication service[13]. In this CBDI report I discussed how BT is working on a long term program which aims to deliver a pervasively used authentication service. The really key point here is that the technical delivery of the service is literally trivial in comparison to the business task. Although BT is partnering with a company highly experienced in authentication and personal data management, they have a huge task to persuade initially businesses and subsequently individuals to change their customary practices in relation to personal identification. This is nothing short of reengineering on a grand scale. What's important to note is that BT is already embarked on this program, long before the technologies are anywhere near mature, and is focusing on business led product design, with the clear intent to converge with the maturing technology.

Web Services Maturity Model

So how do we rationalize where we are in the overall march towards a service oriented world? In our work we advise the importance of pacing activity in line with product maturity, whilst at the same time developing and working towards a longer term vision. The BT authentication program is clearly a good case study.

In Figure 4 we offer a very simple model to assist communication and planning. This model is not intended to be definitive or precise, rather a rough aid to understanding what's going on. For example if the columns don't all match up for all the rows at any point in time don't worry unduly, it's the principle of phased progression that's important, and the need to prepare and manage through it.

Four Phases

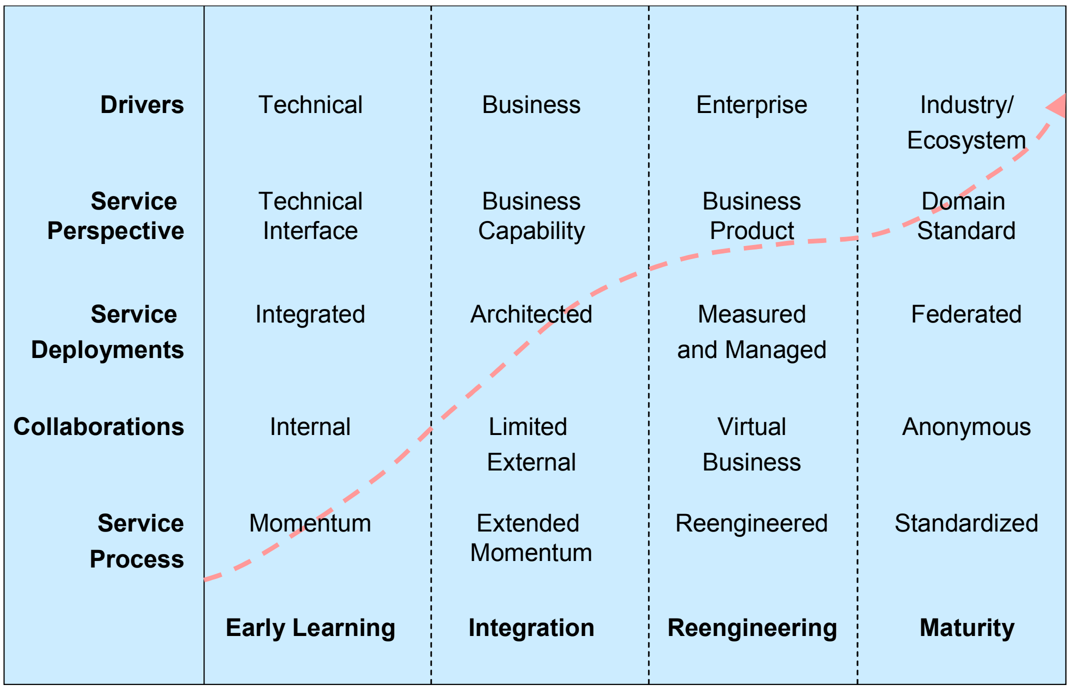
It seems probable that most organizations will go through four major phases in the service oriented environment.

Phase 1 - Early Learning

In this phase it's a technical service world. Early activity is exploratory and mostly about better application integration. Activity is mostly internal and an extension to current activity, and managed under existing processes.

[12] CBDI Best Practice Report - February 2003 - Modeling for SOA
http://www.cbdiforum.com/secure/interact/2003-02/model.php3
CBDI Best Practice Report - April 2003 - Modeling for SOA - Worked Example
http://www.cbdiforum.com/secure/interact/2003-04/model_soa.php3

[13] CBDI April 2003: Product Report - The BT Authentication Service
http://www.cbdiforum.com/secure/interact/2003-04/bt.php3



Drivers	Technical	Business	Enterprise	Industry/ Ecosystem
Service Perspective	Technical Interface	Business Capability	Business Product	Domain Standard
Service Deployments	Integrated	Architected	Measured and Managed	Federated
Collaborations	Internal	Limited External	Virtual Business	Anonymous
Service Process	Momentum	Extended Momentum	Reengineered	Standardized
	Early Learning	**Integration**	**Reengineering**	**Maturity**

Figure 4 - The Web Services Maturity Model

The WSnn protocol efforts in the W3C and more recently in OASIS have now been running for some 3 years, and the all important trio - SOAP, WSDL and UDDI, together with the important WS-I profiles allow a basic level of description and interoperability of messages that establish a first base. We might regard this as completion of a major phase of work, that will permit certain types of application, which might be more easily defined by what they do not permit - for example reliability, transactionality, security etc.

In this phase the predominant service deployments will be:

- mostly internal
- low-risk external
- using existing security mechanisms
- not mission critical
- focused on better application integration

CBDI has reported on numerous case studies that have shown how Web Services can be used to good effect with the basic protocols. For example we might instance Amazon that is very clearly set on reengineering their business and enabling their affiliates to do the same. But these are in a minority. Whilst there will undoubtedly be some new and innovative uses of Web Services, this phase will be best characterized as a period in which many enterprises do their early learning, and some existing applications are wrapped in order to establish some elementary level of service oriented architecture.

It's clear there's a big difference between adopting Web Services and becoming a service oriented organization. Web Services in the end are simply a better form of middleware. Moving beyond that level requires a change in business practices not just technology.

Phase 2 – Integration

In the second phase business drivers will start to become important. This was clear from our survey carried out in February 2003[14], which showed business appreciation of the benefits from

[14] CBDI Web Services Usage Survey
http://www.cbdiforum.com/bronze/webserv_usage/webserv_usage.php3

better, loose coupled architecture. In this second phase SOA is a critical objective, which is justified by greater business flexibility, and creating an application environment where "business" capabilities are exposed as services which can be easily reused and upgraded.

The focus in this phase remains very much on internal activity and external services continue to use current practices and technical architectures for inter company interoperability.

The adoption of SOA does introduce some important delivery process change, as organizations introduce service delivery and management tools and techniques, but in the main this phase is managed by an extension of current practices.

It has become something of a "cause celebre" that the absence of Web Services security has been a significant inhibitor to adoption. However this bears re-examination. Those that have wanted to secure Web Services using what we have defined as first phase protocols, have used pre-existing security mechanisms (SSL, CORBA etc) without difficulty. What the WS-Security protocols bring is heterogeneous message level security, which we suggest is relevant to more sophisticated architectures that also need enhanced guarantees of reliability and availability.

And it seems most likely that the combination of security and reliability will form the basis for a second phase of Web Services adoption that will be characterized by the ability to offer some guarantees around service levels. It also seems likely that this phase will be characterized by some level of business process integration. From a timing perspective it seems highly likely that the BPEL specification will prevail and will, by the early 2004 timeframe, provide a reasonable basis for business process integration. This is corroborated by the primary vendors supporting BPEL that plan to have tools available in this period.

In this phase the predominant service deployments will be:

- Business process oriented
- Project level implementation
- Mostly internal usage
- Based on a more mature understanding of SOA with better separation of layers as BPEL scripting is implemented

Phase 3 – Reengineering

The third phase represents the point in time when the man waving the red flag in front of the automobile is no longer required. When the automobile and the mobile phone are designed for purpose, rather than to replicate prior practices simply because they are de facto conventions.

The term reengineering is perhaps not popular because it has been subject to misuse, but it is highly relevant in this phase where the very purpose of services will undergo change, as well as the delivery technologies and practices.

The primary discontinuity or driver will be the transition to "business product" thinking, where the service becomes the business product. The notion of information technology and systems requirements goes away, because the requirements are fully synonymous with the business product design.

A key pre-condition and driver to this phase is mature measurement and management functionality, as well as ubiquitous external interoperability enabled by mature security and usage profile standards. As we have indicated earlier, it is likely that this phase will coincide with the mature availability of massively simpler (and presumably cheaper) implementation technology which will act as a significant stimulus to action.

For some time now there has been evident something of a disconnect between the ambitions of the major platform vendors and the standards processes. The issue here is management, the protocols which are being worked under the OASIS WSDM committee.

The issue is that enterprise level services will only become acceptable to enterprises when the provider can provide rock solid guarantees of availability, reliability and performance, AND can dynamically manage the environment to deliver that, AND prove in retrospect that the commitments have been met. This will be the major inhibitor to more widespread use and reuse of services. Although there are an extraordinary number of ISV's that have entered this Web Service Management marketplace, the typical enterprise is going to a) see a standards based platform as a prerequisite for implementation and b) prefer to wait and see what the major platform vendors provide. Early adopters will always move with early adopting tools vendors, but the mainstream market will wait.

The current emphasis placed upon on-demand or demand based environments is going to be a critical enabler of the end point service vision. What's interesting is to compare the efforts of the major vendors. In our previous reports on this area, we have characterized the efforts of HP and Sun as primarily about utility computing - creating a flexible technical environment, whereas IBM is increasingly focused on the business opportunities that are enabled by on-demand environments. This increased visibility is going to be essential in our Third Phase, as we have pointed out in a recent CBDI Newswire - Groundhog Day. As one correspondent accurately commented, it's going to be essential to "Convince the main board to address enterprise IS architecture seriously, not as a modeling exercise and not as a means of controlling IT but as an integrated part of business strategy planning."

Our Third Phase is therefore the point in time when service oriented applications are fully integrated with the business from organizational, funding, and product development perspectives, to name just a few.

In this phase the predominant service deployments will be:

- Enterprise level, with common services used right across organizations
- Services implemented as an integral part of business products
- Supported by guarantees and standards based measurement and monitoring systems
- Enabled by a wider selection of available services both inside and external to organizations

Phase 4 – Maturity

One hesitates to write and comment about maturity because it is high probability that by the time we ever reached a mature state that new concepts will have superseded what we are working with today. However in our fourth phase and mature state, services are ubiquitous. Federated services collaborate and create complex products with individual services provided from potentially many providers. Services are designed to support the consumer in their ecosystem, not in a company specific system or service. Many business services such as perhaps the BT authentication service discussed earlier may have become pervasive standards. This process will take considerably longer than the relatively trivial matter of setting protocol standards, which can be carried out by a small group of technologists sitting in isolation on top of a mountain. In contrast business service standards will require huge investment in marketing and process reengineering that may take years to come to maturity.

Timing

The IT archaeologist of the future might be a little puzzled when he or she sees a huge amount of hype and spin about Web Services between 2000 and 2003, which then dies down considerably. What's happening here is that the industry actually didn't over hype the "potential" for Web Services, but what they "did" do, was to set the expectation that the vision would be delivered in a relatively short timescale. The reality is very different. We are currently in Phase One as we have defined it. Many organizations will only enter that phase this year and next. Phase Two will commence in 2004 and run through 2007. Phase 3 will commence sometime perhaps late 2005, and run and run.

It is also important to recognize that the standards processes as we see them today are not the end of the line. As with every standards effort, these are going to continue. One of the most important continuing efforts, and conversely the weakest of the current standards efforts, is the whole area of process choreography. Whilst BPEL is considerably better defined than the competing specification WSCI, it does not deliver a comprehensive view of the contract between the provider and consumer.

The importance of contracts should not be underestimated. It is very difficult to adopt an on demand approach (to either computing resources, or business processes) without having a very clear and precise way of expressing machine readable contracts. This is a major failing, because without it, there is going to have to be incremental communications between the collaborating parties. Widespread reuse of services will only occur when the consumer is given a comprehensive contractual view that documents all the behaviors and obligations, in other words a specification model of the externalized perspective of the offered service. This is perhaps the next big challenge, and as yet this requirement is not yet even comprehended by some major players in the modeling tools space.

Summary

So don't be surprised when you hear the major vendors de-emphasizing Web Services. They know they have over reached themselves on this issue, and that they need to redirect attention for a time. On demand, demand computing, efficiency and cost reduction are all good themes that play to today's economic circumstances. This is where the vendors will address their efforts, and not unreasonably establish the necessary operating environment for a world where use of services is pervasive.

For the same reasons the vendors are characterizing Web Services as "just another form of middleware". They realize they have to adjust their messages for a while, and play down the strategic importance of Web Services, until they are ready for mainstream deployment. So focusing on just another form of middleware is resetting expectations to be relevant to applications that can reasonably be implemented during our Phase One and Two.

It's important to understand that this is a longer term game we are playing. Web Services is merely a phase in a longer running process which will eventually deliver a comprehensive service oriented environment. It's going to take time, but there's much that can be cost justified today, which equally contributes building blocks to the overall project.

The primary message in this report is the criticality of managing the process of evolution. Those businesses that sit back and allow technology matters to drive their use of services will almost certainly fail to survive what looks like a profound change in business practices, which will occur in the second half of this decade.

What's needed is a balanced approach to building the technology with today's tools, recognizing that these will certainly be superseded in due course. The most important issues are establishing the loose coupled SOA architecture that hides the implementation and allows transparent technology upgrade, while the reengineered product planning and process management practices deliver on innovative new business services that provide real competitive edge in the right timeframe.

Roadmap Actions
Establish a service oriented strategy - Establish organizational consensus on the overall plan for transitioning business and technology environments, synchronizing relative maturity of capabilities and applications
Identify business opportunities - look at mundane business processes and services in a new context, plan support at the service ecosystem level
Set up long term R & D and service oriented projects - don't wait until the technology and standards are available, business development always takes longer than IT
Review Technical Strategy - augment technology strategies to prepare for service orientation

ROI – The Costs and Benefits of Web Services and Service Oriented Architecture

Having difficulty persuading your colleagues or business sponsors of the merit of adopting Service Oriented Architecture (SOA) or using Web Services instead of some existing mature technology? Need to understand what the costs will be, not just the potential benefits? Then this section should help.

Why Web Services?

After a couple of years of hype you would imagine everyone is now familiar with the justification for using Web Services. And yet, on many occasions the industry still struggles to clearly put across the benefits of Web Services and articulate precisely how they differ from existing technology solutions.

The issue is that superficially much of the benefits that are attributed to Web Services have also been claimed by pretty much every new technology over the past n years. You might realistically argue that the IT industry is well known for hyperbole and exaggeration, and that reality generally falls short of expectations. Things like "improving business agility", "reducing time to market", etc are still valid - but not entirely new. Why is Web Services going to deliver this time? Consequently, there is a real need to be much more precise about the specific cost savings and benefits for both business and IT that can reasonably be attributed to Web Services.

This is not so surprising because Web Services have been a technology led paradigm, and early usage has often been in the area of internal integration where benefits are quite straightforward. Grand visions of everything being connected in dynamic real time scenarios are one thing, but most organizations have more mundane problems to solve. And so we seemingly get stuck in the middle, not always knowing which aspect of Web Services to highlight. Too visionary and the audience gets scared (early adoption = risk), too mundane and they get bored (their existing solutions suffice)

Also, the constant evolution of Web Services means that the benefits keep evolving too. What started out as a simple distributed computing solution can now be applied to wider range of connectivity scenarios. So just what are Web Services now? Is it just a better, cheaper, faster (BCF) interface for existing EAI, distributed computing, EDI or other scenarios? Or is it more than that? Of course there should be every reason to use Web Services if they are truly BCF in these scenarios, but this is largely an IT centric message. And a frequent response is that existing solutions in these scenarios are mature and largely get the job done - so why convert to Web Services?

Perhaps because the ideas are more abstract, we often see the industry failing to put across what are some of the key differentiators of Web Services. For example the richness of the service specification that can be conveyed in WSDL and emerging business process orchestration standards that enable self describing services and self discovering applications. If you just need your developer to connect A to B, then EAI might suffice. But if you want A to dynamically find B, and later switch to C, without developer intervention, and regardless of the technology that A, B and C use, then Web Services is the only solution.

So, what then are the benefits and motivations for using Web Services? And what are the precise features that deliver them? Original definitions of Web Services are now outdated as they were too narrow in their viewpoint, which was typically focused on an RPC centric, object/component oriented view of the world. So we offer up a broader definition, which also now recognises that Web Services are not necessarily anything to do with the web:

"Web Services provide a simplified mechanism to connect applications regardless of the technology or devices they use, or their location. They are based on industry standard protocols with universal vendor support that can leverage the internet for low cost communications, as well as other transport mechanisms. The loosely coupled messaging approach supports multiple connectivity and information sharing scenarios via services that are self describing and can be automatically discovered."

To understand how both businesses and IT departments benefit from Web Services in detail, let's dissect this statement and consider it line by line.

1. A Simplified Mechanism To Connect Applications Regardless Of The Technology Or Devices They Use, Or Their Location

Web Services will be ubiquitous. Applications running anywhere, on any technology or device will have a Web Services capability available to them. As such, the applications of customers and business partners will be able to participate in an organizations business process, in real time.

Business Benefit – Wide Area Business Process Efficiency	**IT Benefit – Cost Reduction**
Potentially improves business process efficiency by reducing cost and particularly time to connect applications.	Lowers the cost of connection.
Increases the feasibility of real time, remote access to core source of information (owner) which provides current information to a process.	Reduces complexity of integration.
Enables real time business, and straight through processing.	Delivers platform and technology independence
Customer, partner and employee enablement.	

2. Based on Industry Standard Protocols with universal support

Previous solutions are typically proprietary, and even if so called ‘standards’ have failed to reach universal adoption.

Business And IT Benefit – Cost Reduction And Choice
Both Business and IT gain traditional “open standards” benefits
Not locked into proprietary technology
A wide choice of suppliers
Reduction in technology costs through commoditisation
Increased quality through competition on implementation

3. Leverages the Internet for low cost communications

The high cost of private networks, coupled with the cost of proprietary EDI/B2B solutions has been a barrier to entry for many organizations. Though big participants can justify this, many industries have thousands of small organizations who cannot. Similarly organizations like retailers with an extensive branch network, can more easily enable them to participate in real time communications.

Business Benefit - **Lowers The Barrier To Entry**	**IT Benefit –** **Simplified Middleware**
Available to all sizes of organization, and individuals	Same technology can be used for both external and internal connections
Low cost means thousands of small partners and suppliers, or a branch network, can now be integrated	Leverage ubiquitous Internet protocols and infrastructure
Supports globalisation. Integration of geographically dispersed organizations	

4. As well as other transport mechanisms

Whilst the Internet provides a ubiquitous, low cost transport for Web Services, the "web" in Web Service is now misleading, as bindings to other transport mechanisms are available which may be more appropriate to internal usage or private networks where higher speed and more robust connections are available. Though this might seem to contradict point 3, it nevertheless reflects that for some scenarios the Internet will not be ideal, and that Web Service protocols that will improve the reliability of Internet based communications are not yet mature. The key benefit, primarily to IT, is that it now provides choice of transport.

IT Benefit – Can choose transport most suitable to need
Leverage existing transport infrastructure
Deliver Web Services over reliable, robust, fast transport mechanisms
Options for both internal and external Web Services

5. Loosely Coupled

Previous connectivity approaches required the same technology at each end of the wire. For example, even though EAI adaptors enabled different applications to connect to each other, it still required the same proprietary EAI technology as a wrapper around each application. Focusing on XML protocols, Web Services describe the connection, not the technology at either end. Loose coupling is not just a technology issue however, but a key aspect of service design.

Business Benefit – **Agile Relationships**	**IT Benefit –** **Reduce Cost Of Maintenance**
Makes it easier to change or add partners.	Lower cost of maintenance
Facilitates M&A activity	Reduced impact of change
System change is not a constraint on business change	Facilitates reuse of existing assets

6. Supports Multiple Connectivity and Information Sharing Scenarios

Today organizations use different technologies for distributed computing, EAI, EDI, B2B, Websites, Portals. This results in n times the products, tools, skills and cost. Web Services provides an opportunity to radically reduce this by supporting these different scenarios with the same basic protocol stack.

IT Benefit – Cost Savings Through Consolidation
Broad applicability reduces the number of different products, tools, skills, etc required
Consistent approach in all scenarios
Common infrastructure can be leveraged across all scenarios – e.g. security

7. Self Describing

The time taken for developers to properly understand how to use an existing interface – particularly when it is external to their own projects – slows down the time that new connections can be established. Web Services provides a much richer specification of the service compared to previous technologies, which can be accessed programmatically.

Business Benefit – Time To Market	**IT Benefit – Shortened Development Cycles**
Improves time to market as connections to partners and customers can be made faster, even dynamically.	Reduces development effort as consumption of service is largely automated.
Makes it easier for partners to do business with you.	Reduced impact of change. Response to changes can be automated.
	Services can be consumed dynamically without developer intervention.

8. Automated Discovery

Provides a mechanism for discovering Service Providers, which can be automated.

Business Benefit – Faster Extension Of Ecosystem	**IT Benefit – Cost And Time Saving Through Automation**
Makes it easier for customers to find you and your services	Reduces or removes development effort to support new connections
Makes it easier to find new partners and their services	

Why Web Services Should Work

Déjà vu? Heard it all before? Here are some reasons why Web Services should work where prior approaches have failed

1. **Universal Support.** Previous de facto, or de jure interoperability standards usually lacked participation by one or more key vendors.
2. **Protocol not Platform.** It doesn't require the adoption of a common platform, just adherence to the standard protocols.
3. **Low Product Costs.** Sure vendors will want to sell products that optimise the Web Service experience in some way, but the basic needs are going to be supported essentially for free as part of the platform or as an adjunct to some existing product. Nor are they expensive to adopt in term of development effort. Everyone can play this game without major investment.
4. **Evolutionary.** Long term, Web Services will likely usher in some revolutionary new ideas that force the replacement of existing systems. In the meantime consider Web Services as evolutionary and use them to leverage your existing applications and infrastructure rather than requiring rip and replace - another low cost benefit.
5. **Business Oriented.** Web Services don't just appeal to the technician, but directly address real business needs of today. The business didn't care about OO or CBD, but should care about Web Services, particularly where they directly reflect some meaningful business concept, and especially where they are being exposed external – and hence are a reflection on the business.

Why SOA?

Web Services are not a silver bullet. Like most technologies, it is only by ensuring that business requirements are properly understood and their application is carefully designed, that the benefits claimed are truly delivered. It is very easy to deliver bad Web Services.

Whilst Web Services remove many of the technology constraints of communication between applications providing flexibility at the implementation layer, the business agility that is promised is more a factor of Service design than protocol adoption.

As such SOA should be thought of not just as a way of designing and documenting an "Architecture of Services", showing their relationships, dependencies, etc., but also a discipline by which we ensure that those Services are the right Services, delivered at appropriate levels of granularity, abstraction and generality that makes sense to both Service Provider and Service Consumer, reduces the effort (particularly on the consumer) to use a set of services to perform a particular objective, and truly minimises the impact of change allowing Service consumers to switch providers and Service providers to switch implementations.

We believe that SOA disciplines will become vital in delivering external Web Services where agility and flexibility is required (by both provider and consumer), and for Enterprise Wide Web Services where broad applicability must be ensured to enable reuse.

The Costs of Web Services and SOA

Many of the benefits outlined above imply reduce IT costs resulting from the adoption of Web Services. What, if any are the additional costs of using Web Services and adopting SOA?

Thanks to the universal adoption of Web Services by vendors, much of the software infrastructure required effectively comes as part of the regular upgrade cycle of existing products that most organizations will already have. This is not to say that organizations will not take the opportunity to decide whether they need to change products or vendors in order to obtain software better optimized for Web Services, or acquire additional products that might make them more productive. For example, some organizations might introduce new capabilities they have not used before such as business process orchestration/workflow products that now support Web Service protocols.

However, the bottom line is that most of the essential Web Services capability can be acquired via software upgrades that also contain other useful functionality, and as such comes at effectively zero or little additional cost in terms of software.

The Cost of SOA

Delivery of the Web Services themselves in terms of the necessary protocols if largely hidden from developers by tools, and is unlikely to be an overhead in terms of programmer productivity.

As such, much of the development overhead of delivering Web Services will come in the analysis and design phase to ensure that where required Services are

- Properly abstracted away from the implementation to deliver flexibility and agility
- Sufficiently generalised to enable enterprise wide applicability

Performance Overheads

Whilst there are ways of optimising the use of XML, many Web Service scenarios will involve extra process steps that will likely add some performance overhead to the overall process. That said, existing alternative integration options such as EAI and B2B have similar performance profiles in comparison to a point to point connection between two applications using the same (proprietary) technology. Some possible sources of performance overheads will be,

- Service Wrappers and Facades around existing systems
- Transformation in and out of XML
- Dynamic Web Service Management
- Federated Security
- Services routed via external intermediaries

Of course these should all be seen from the perspective of the benefit they bring, not just the overheads they incur.

Software costs and Product Acquisitions	
Software Upgrades to support Web Service protocols	➔ Likely to be part of regular upgrade cycle – not specific to Web Services
Web Services Management and Web Service "Utilities"	↑ Acquire or build ➔ Some capability will be delivered as part of server platform upgrades
Private UDDI	↑ Software essentially free, but still requires deployment management. May require dedicated server(s)
Security Infrastructure	↑ upgrades to support federated security, improved identity management, etc
See "Assembling the Web Services Infrastructure" http://roadmap.cbdiforum.com/reports/infrastructure/	
Organizational	
Skills	↑ SOA Skills will need to be acquired, learned ➔ Web Services technology largely automated and hidden from developer
Roles	↑ New roles with new disciplines

Development	
Enterprise SOA	↑ Ensuring Services are applicable Enterprise-wide
Abstraction	↑ Ensuring Services are properly abstracted away from the implementation
Collaborative development	↑ Jointly agreeing and designing Services in collaboration with other participants (providers and consumers)
Service Wrappers for existing systems	→ Often these will come at little cost or effort where vendors are Web Service enabling the underlying platform, or packaged application, as part of their upgrade cycle ↑ However, careful design is needed to ensure abstraction
Testing	↑ Web Services should be real “black boxes”, requiring more diligence in testing as the implementation is not available for inspection.

Table 8 - Costs of Web Services and SOA

Roadmap Actions	
Business drivers	Align Web Service benefits with business requirements.
Strategy and policy	Identify business opportunities and support to Technical Strategy Identify areas for cross organizational coordination, establish reuse strategies and mechanisms for cross organizational policy setting, projects support, decision making
Infrastructure	Justify infrastructure investments on the basis of clear benefits
Process and governance	Incorporate WS benefit assessment into life cycle
Application (areas of usage)	Be clear in the justification for using Web Services rather than any other approach
Organizational change	Ensure that education covers not just the what and the how, but the why of Web Services Ensure that SOA is seen as an additional discipline with it’s own costs and benefits, and is not an automatic by-product of using Web Services

The Web Services Protocol Stack

Abstract: This report assesses the status of various Web Service protocols and suggests a timeline for their adoption and relevant roadmap actions. It provides a useful reference and links to all the numerous protocols currently proposed or in the standards process

Web Services are a set of protocols based on XML (Extensible Markup Language). Many readers will be familiar with the following base protocols that formed the initial specification for Web Services.

- Simple Object Access Protocol (SOAP) - defines the runtime message that contains the service request and response. SOAP is independent of any particular transport and implementation technology.
- Web Services Description Language (WSDL) - describes a Web Service and the SOAP Message. It provides a programmatic way to describe what a service does, paving the way for automation.
- Universal Discovery, Description, Integration (UDDI) - UDDI is a cross industry initiative to create a standard for service discovery together with a registry facility that facilitates the publishing and discovery processes.

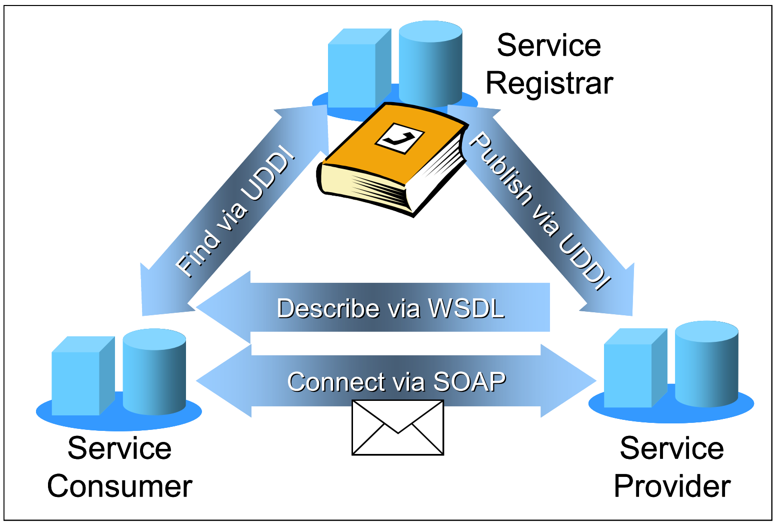


Figure 5 - Base Web Service Protocols

These have effectively become de facto standards, with effectively universal acceptance and widespread implementation by vendors. Figure 5 shows the way their application is typically illustrated.

These base protocols have enabled many companies to put straightforward Web Services into production. However, to improve the security and reliability of Web Services and to address more complex business scenarios, a wide range of additional protocols have since been proposed. Some of these have since been merged with others or morphed into new proposals. The current proposals are illustrated in Figure 6. Further detail and links to the various specifications is provided on the Roadmap website[15] which is kept constantly up to date with their evolving status.

Web Services Architecture

The additional protocols have been proposed within the context of a modular framework that would allow,

[15] http://roadmap.cbdiforum.com/reports/protocols/summary.php

Developers to only use the modules needed for their Web Services. Each module can be lightweight and not overburdened with irrelevant syntax.

Each module to evolve in isolation

The W3C have since formed a Web Services Architecture Working Group[16] and Figure 6 is loosely based on the architecture in their current draft. Microsoft has also promoted their Global XML Web Services Architecture (GXA), but this is essentially the same and should not be misunderstood as a Microsoft proprietary alternative.

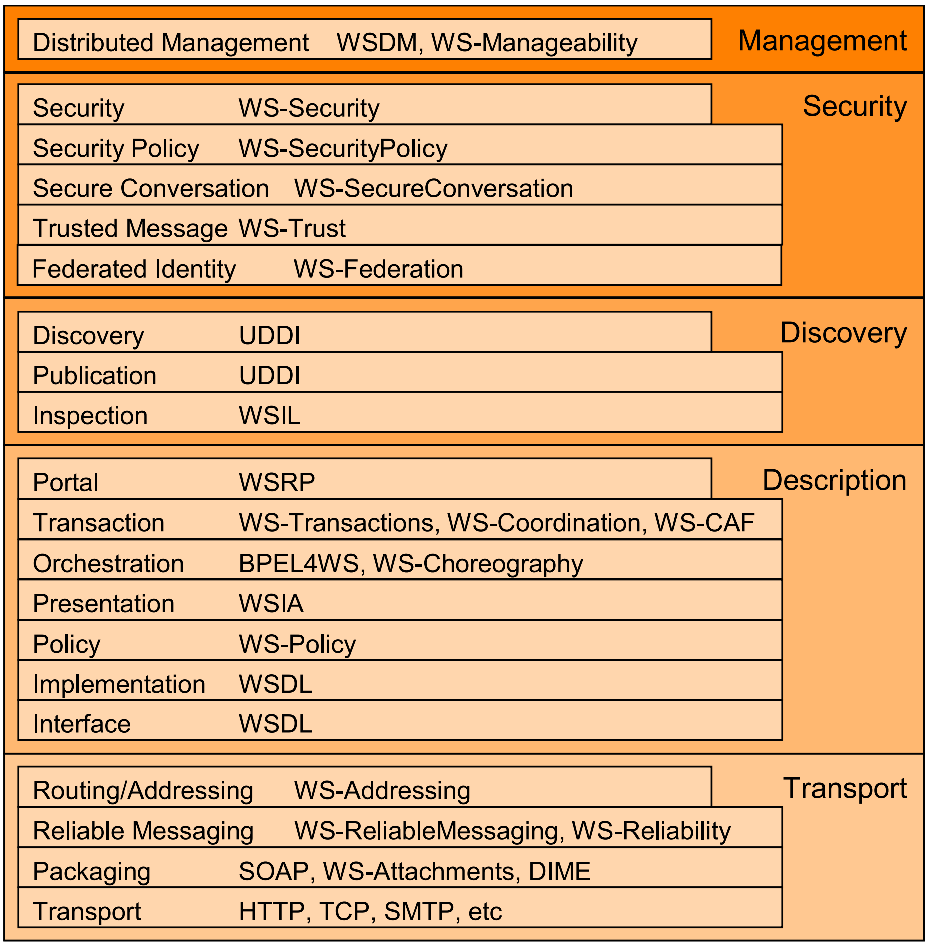

Distributed Management WSDM, WS-Manageability
Management
Security WS-Security
Security
Security Policy WS-SecurityPolicy
Secure Conversation WS-SecureConversation
Trusted Message WS-Trust
Federated Identity WS-Federation
Discovery UDDI
Discovery
Publication UDDI
Inspection WSIL
Portal WSRP
Description
Transaction WS-Transactions, WS-Coordination, WS-CAF
Orchestration BPEL4WS, WS-Choreography
Presentation WSIA
Policy WS-Policy
Implementation WSDL
Interface WSDL
Routing/Addressing WS-Addressing
Transport
Reliable Messaging WS-ReliableMessaging, WS-Reliability
Packaging SOAP, WS-Attachments, DIME
Transport HTTP, TCP, SMTP, etc

Figure 6 - The Current Web Service Protocol Stack

CBDI Assessment

Additional Protocols Required

Taking all the proposals in Figure 6 into consideration, the set of protocols required for secure, reliable 'Enterprise' Web Services is largely complete. Areas not fully addressed are

- Management. The OASIS WSDM Technical Committee has only just been established. As yet, the full scope of the protocols they will propose is not clear. The intention is primarily to provide protocols to enable the run-time management of Web Services and enable Web Service platforms to feed relevant information to traditional Systems Management tools.

[16] http://www.w3.org/2002/ws/arch/

- Service and Business Level Agreements. These are identified by the W3C Web Service Architecture working group as part of the description layer, but as yet no proposals have been made in this area.
- WS-Security. The specifications for some elements of the WS-Security architecture have yet to be published. These are WS-Authorization and WS-Privacy.

Alternative Proposals

The degree of industry consensus on Web Service protocols has been significant. Though alternative proposals have been made in some areas, the formation of an appropriate working group in either W3C or OASIS has usually seen the subsequent convergence of all interested parties.

However, there are currently alternative proposals, namely in the areas of Reliable Messaging, Orchestration, and most recently Coordination and Transactions. The alternatives reflect an IBM/Microsoft led initiative on one side, and one led by Sun/Oracle on the other. W3C have kicked off a WS-Choreography (orchestration) working group and OASIS a Reliable Messaging TC without IBM or Microsoft's participation. Subsequently, IBM, Microsoft and their partners formed the Web Services Business Process Execution Language TC at OASIS to further their BPEL4WS proposals as a direct alternative to W3C WS-Choreography. Recently, Sun, Oracle and others have published the WS-Composite Application Framework as an alternative to the WS-Coordination proposal from IBM, Microsoft, and BEA.

We judge the IBM/Microsoft proposals to be technically more advanced than the alternatives, together with support from key industry leaders in each space, including a number of companies such as SAP and Tibco who originally backed WS-Choreography but have now also joined WS-BPEL. Given that, it is still early in the standardization process for protocols in these areas, and we anticipate that ultimately there will be convergence and consensus as in the other areas.

There is also some overlap with the ebXML initiative. ebXML uses SOAP at the transport level, but has its own registry and orchestration. Though ebXML is an approved, robust standard its applicability is far narrower than Web Services. As an evolution of EDI, it primarily addresses B2B only. As such we believe the Web Service protocols that are designed to address multiple requirements will prove more valuable in time and that ebXML will probably evolve to adopt additional Web Service protocols as they mature and are approved.

Standardization Process

Though the proposal of various Web Services protocols has been a fast moving area, their transition into actual open standards is inevitably much slower. There are only a few protocols that have, or a close to completing the standards process proper. Some key proposals have yet to be submitted to any standards body. We advise continuous monitoring of what are currently the two main standards groups involved in Web Services,

- World Wide Web Consortium (W3C)– www.w3c.org
- Organization for the Advancement of Structured Information Standards (OASIS) – www.oasis-open.org

The CBDI Web service protocols summary table[15] on the roadmap website indicates the current status of the various protocols in the standards process

WS-Interoperability (WS-I)

WS-I is an open, industry group that was formed in 2002 to promote Web services interoperability across platforms, operating systems, and programming languages. Though this would appear to be the basic premise of Web Services and the role of standards bodies, WS-I still has a useful role to play, for example,

- Standards specifications are always open to interpretation to some extent. WS-I will provide guidelines and tools to help measure the conformance of various implementations, and to enable their interoperability
- As standards evolve, there is a need to understand what different versions might interoperate

- Publishing interoperability profiles to reflect the above, one of the key deliverables of WS-I.

Adoption

An assessment of the timescales for adoption of these protocols is provided in Figure 3. Given the alternatives that have been proposed for some protocols, we judge the future of those not currently backed by the groups containing IBM and Microsoft to be uncertain.

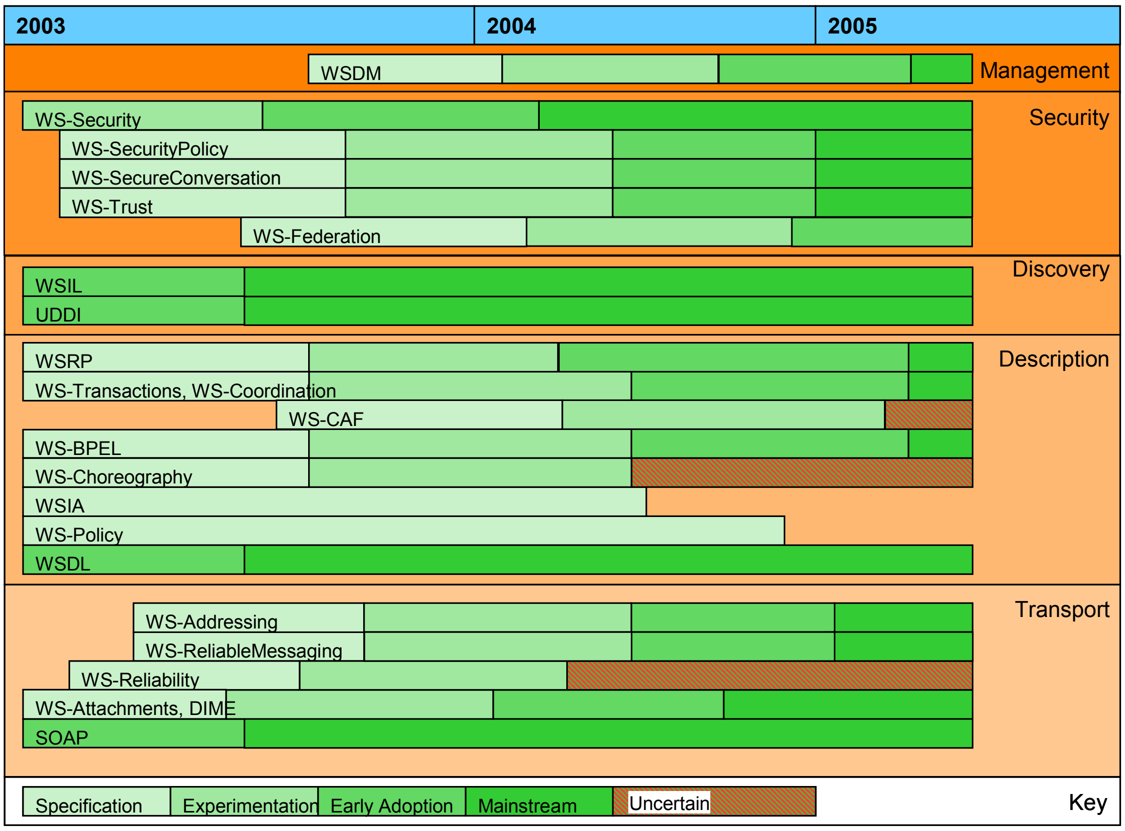


Figure 7 – Likely adoption rate of Web Service Protocols

- *Specification – Exists only as draft specification. Any usage requires hand coding.*
- *Experimentation – early implementations provided by vendors permit experimentation, but are not recommended for production use. (e.g. technologies available from IBM Alphaworks do not support production use)*
- *Early adoption – More robust implementations available and protocol well into standards process, encourages production usage by end user organizations*
- *Mainstream – standard ratified, or wide scale de facto adoption*

Roadmap Actions

Apart from infrastructure and tools vendors, and early experimentation, organizations should avoid handcrafting the use of Web Service protocols wherever possible. It should not be necessary for developers to learn the low-level XML syntax of Web Services, delegating the generation of it instead to the infrastructure products and development tools.

Organizations should establish a policy for compliance with standards, paying particular attention to evolving versions and using WS-I profiles wherever relevant.

Roadmap Actions
Monitor progress of protocols through key standards bodies
Establish policy on protocol usage
Adopt protocols as WS-Profiles become available to ensure standards based interoperability. Create local profiles only where necessary, and plan to upgrade to WS-I as they are published
Coordinate use of protocols to ensure consistent implementation of versions and profiles. Publish best practices
Plan for phased implementation of emerging protocols with local extensions where necessary
Wherever possible wait for implementation of protocols in products

Assembling the Web Service Infrastructure

With each technology shift, organizations need to re-examine their infrastructure to determine how it will support the new requirements. Web Services now raise the question again and in this report we examine the issues facing organizations and some of the options available, as well as exploring a phased approach to infrastructure upgrades.

Introduction

With each technology shift, organizations need to re-examine their infrastructure to determine how it will support the new requirements. With regard to Web Services, organizations need to consider such issues as,

- What elements of their existing infrastructure need upgrading to provide specific support for Web Service protocols? Is XML capability enough? Or does it need specific support for Web Service protocols?
- To what extent will their existing infrastructure built to support the Internet suffice?
- Are there new classes of tools required?
- Is the same infrastructure capable of supporting both internal and external Web Services?
- How far into the infrastructure do Web Services need to penetrate? Will a new layer or façade on the existing infrastructure suffice, or must the whole infrastructure be upgraded?

There are several elements to the Web Services infrastructure, as listed in Table 9. The requirements for this Infrastructure may be served by more than one type of software component/product and organizations need to carefully examine how their full needs are going to be addressed. For example, a web/application server may contain some measure of all these requirements, but at the same time may not be as comprehensive as a dedicated product (perhaps from an ISV rather than a platform vendor) in any one of these domains.

At one extreme, some organizations with a small number of straightforward Web Services may feel their requirements are addressed by little more than a Web Service capable web server such as Apache AXIS and a few well chosen scripts. Whereas larger organizations, delivering numerous business critical Web Services with demanding SLAs, will see an overall benefit from adopting several dedicated infrastructure components.

As with other technology shifts, many start ups or existing ISVs are delivering new products dedicated to Web Services. At the same time, the major platform vendors are also delivering useful Web Service capability as upgrades to their platform and middleware products. In Table 10 we examine the support for Web Service protocols in infrastructure products.

Infrastructure Requirement	Likely Source Component	Comments
Deployment Run time handling of Web Service protocols	Web/Application Server	Built into Servers such as Microsoft .NET, J2EE1.4, Apache AXIS, BEA WebLogic
Management Run time management of services, messages, users, etc	Web/Application Server Web Service Management (WSM) Systems Management	Elements of both passive and active management will be contained in the underlying platform and the web/application server. However dedicated WSM products allow you to abstract and centralise management away from the multiple (heterogeneous) applications.
Security Identification and authentication of participants, protection against cyber attack	Web Services Firewall Identification Services Web Service Management	Though elements of security will be built into other components, there is merit in addressing security via separate, stand-alone components
Orchestration Run time workflow/process control	Workflow/Orchestration engine	Look for products based on emerging Web Service protocols such as BPEL4WS, WS-Choreography
Protocol Creation Delivery and description of Web Service in terms of Web Service protocols	Web/Application Server Development Tools	.NET and J2EE 1.4 will automatically create the necessary protocols
Development Build/assemble implementation of above	Development Tools Workflow/Orchestration Tools	Development tools will become less focused on the creation of Web Service protocols, and more focused on service design, and solution assembly
Publication and Discovery	UDDI Directory	Public and Private directories can support both external and internal use respectively

Table 9 - Essential Infrastructure Components

Existing Infrastructure	WS Protocols support in current Infrastructure	Use of new WS aware products	Outlook
Routers and Firewalls	☒ Little or limited	Supplement with new XML and WS Firewalls and routers	Longer term expect WS support embedded in hardware routers and firmware Implementation of WS-Addressing for SOAP routing
Directory and Security Servers	☒ Little or limited	Supplement with private UDDI directory	UDDI layered on LDAP
Application and Web Servers	☑☑ Already upgraded		Some WS infrastructure in other categories will be increasingly embedded in the Web/app servers
OLTP, ORB, and MOM	☑ Being upgraded		Though SOAP aware, these products will need some further upgrading to support emerging standards such as WS-Transactions, etc
EAI Tools	☑ Being upgraded	Could supplement with broker capabilities in some WS Management tools	WS will reduce need for EAI adaptor capability. EAI remains useful to wrap existing systems as WS
Orchestration Engines	☒ Only technology previews currently		Products based on emerging BPEL and/or WS-Choreography protocols
Systems Management	☒ Little, or limited	Supplement with dedicated WS Management	Traditional SM tools will add WS capability WSDM standards emerge in 2004
Development Tools	☑☑ Already upgraded		Although upgraded to support basic WS Protocols, few support SOA principals

Table 10 - Infrastructure Support for Web Service Protocols

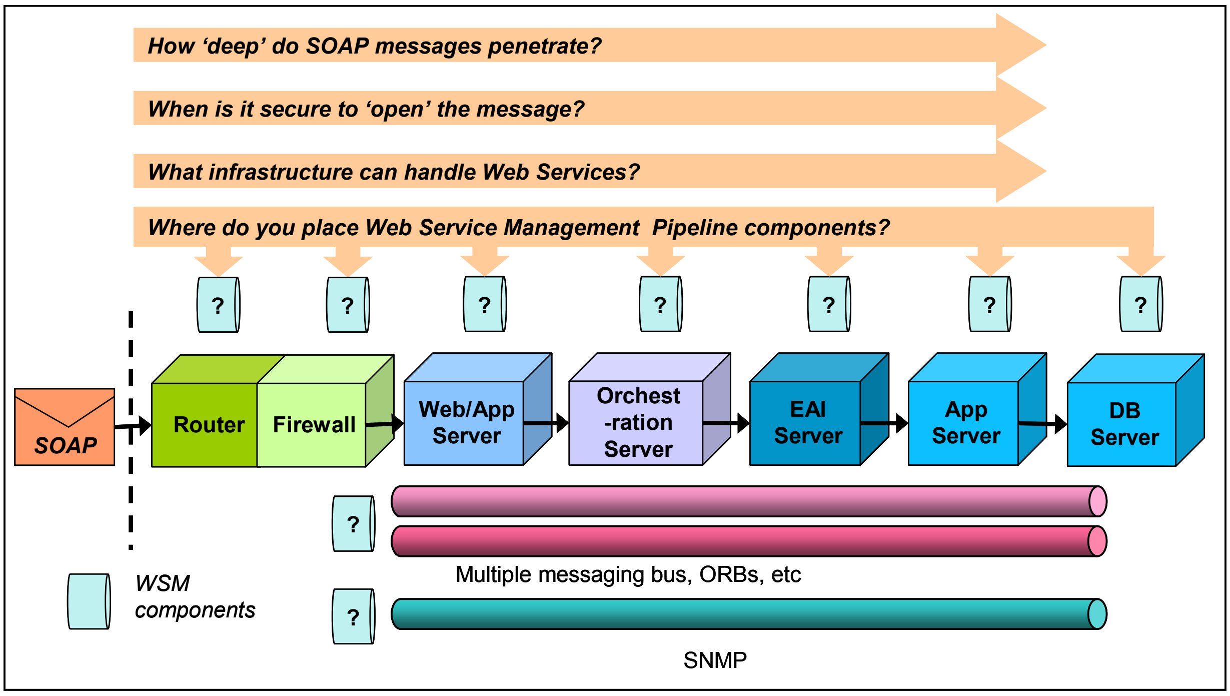
Web Service Infrastructure Architecture

There are a number of deployment options for Web Services Infrastructure.

As with other infrastructure deployment, for larger organizations we believe the ideal would be to put a comprehensive Web Services infrastructure in place that can for example,

- Be reused by any application or service developers across the organizations, so they don't have to reinvent the wheel in each project
- Ensure consistency of approach, management, and security across the organization
- Where relevant operate at the business service level, abstracted away from the (multiple) back end implementations

Like the collaborative SOA approach that organizations are encouraged to implement in their applications using Web Services, then the Web Service infrastructure itself will also follow suit. This will remove some the need for a centralised implementation or to funnel Web Services through specific infrastructure server as components of the infrastructure can themselves collaborate via Web Service protocols.



How 'deep' do SOAP messages penetrate?
When is it secure to 'open' the message?
What infrastructure can handle Web Services?
Where do you place Web Service Management Pipeline components?
?
SOAP
Router
Firewall
Web/App Server
Orchest -ration Server
EAI Server
App Server
DB Server
Multiple messaging bus, ORBs, etc
SNMP
WSM components

Figure 8 - Web Services and Physical Pipeline

One key architectural question is how far should Web Services 'penetrate' into the typical enterprise infrastructure. Now that vendors are Web Service enabling a diverse range of products, as illustrated in Figure 8 a SOAP message might pass through, or be directed to, several physical server types and across multiple networks within the enterprise before reaching its final destination.

At a minimum, an organization could halt the SOAP message at the Web Server, and convert it to existing infrastructure protocols to forward the message on to the appropriate internal system. However, besides obvious benefits of platform independence and the ability for the SOAP message to more easily navigate a heterogeneous environment there would be a number of other benefits in using Web Service protocols across the wider infrastructure for end-to-end message flow. These, together with some of the downsides of adopting this approach are considered in Table 11.

Pros	Cons
Provides platform and transport independent messaging infrastructure	Need to upgrade each of the infrastructure elements listed in tables 1 and 2 with support for Web Service protocols. And eventually every instance.
Use WS-Security to maintain the integrity of a message right until its ultimate destination, keeping it secure inside the firewall, not just outside.	Support for emerging (and more complex) Web Service protocols will take longer to apply to existing infrastructure elements than basic SOAP/WSDL support. Some might not be upgraded by vendors
Use WS-Addressing to route messages to specific servers and applications.	
Development and assembly of Internal applications and components can benefit from protocols such as WSDL, and publication via a private UDDI registry	Fragmentation of development assets that target each of the infrastructure elements. Needs careful co-ordination
Remove need for overhead of transformation to existing protocols – and potential errors	Performance of XML based infrastructure may not be as optimised as existing protocols.

Table 11 - Pros and Cons of Deploying Web Services Infrastructure Enterprise-Wide

Web Service Management

One new category of infrastructure introduced is that of Web Services Management (WSM). WSM complements existing systems management software by operating at the Service level. This also raises an opportunity to monitor and manage at a level meaningful to the business instead of the low level operations, providing Web Services are delivered at the appropriate level of granularity and abstraction.

In the near and mid term organizations should consider adopting tools on offer from a number of ISVs. Ultimately we expect existing systems management tools to provide WSM capabilities, though this does not mean leading WSM vendors will not endure.

We will look at the roadmap for WSM in a further report. In the meantime, the capabilities that organizations should be looking for in WSM tools are considered in our Business Services Server report[17]

Stages of Web Service Infrastructure Deployment

Deployment of Web Service enabled infrastructure is likely to happen in 3 stages as illustrated in Figure 9. In some cases, organizations may expose limited external Web Services as their first use, though as explained elsewhere it is becoming more common for organizations to adopt internal usage first. Actual infrastructure elements that need to be upgraded or added at each stage will of course vary from organization, which is explored in Table 12.

[17] Business Services Service, CBDI Report (FOC), http://www.cbdiforum.com/bus_services.php3

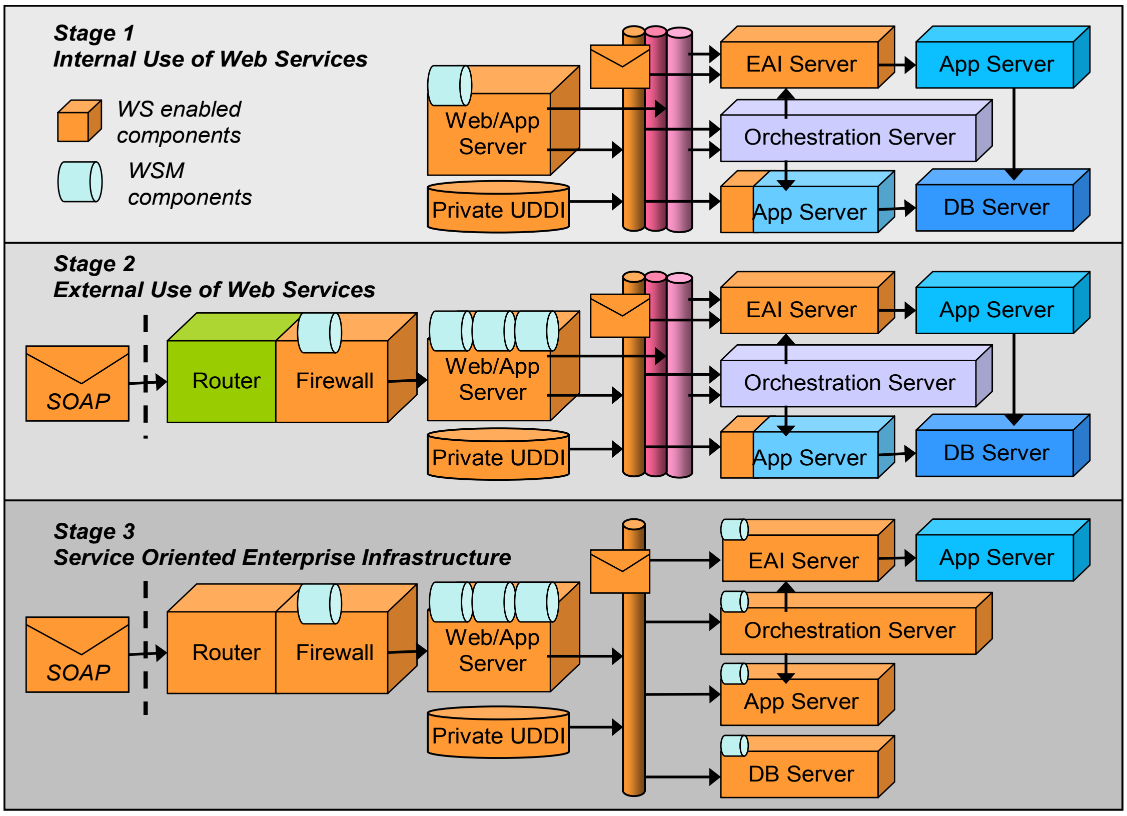


Figure 9 - Web Service Infrastructure Deployment Stages

Upgrades	Comment
Stage One – Internal usage – limited upgrade to key infrastructure where required	
Web/App server	Likely platform for new cross functional business systems requiring WS
EAI server	Front end current non-WS enabled systems. Some orchestration.
Other App/DB servers	Upgrade current or ‘legacy’ app, OLTP, DB servers where some limited native WS now provided. Otherwise use EAI to access.
UDDI Registry	Private UDDI registry for internal use
Web Service Management	Basic monitoring of internal WS to provide performance and usage statistics, and raise alerts to basic problems
Message Orient Middleware and/or ORB	Possibly upgrade existing MOM to provide reliable messaging. Similar upgrade ORB to integrate existing object infrastructure
Stage Two – External usage – main impact security and additional WSM to manage SLA	
Web Service Management	Add modules for SLA Management, User Management, Access control, and Message Management to support External usage
XML Firewall	External usage requires XML/WS level firewall and security

Stage Three – Enterprise Wide SOA infrastructure - underlying platforms/ transports transparent. Enables shift to On Demand Operating Environment by "virtualizing" the back end implementations.	
Web Service Management	WSM components deployed to individual servers. Eventually compliant with emerging WSDM standards
XML Router	Compatible with WS-Addressing
Orchestration Server	BPEL4WS, and/or WS-Choreography compliant business process orchestration
Web/App/DB Server	Upgrade where full WS capability provided, including emerging enterprise standards

Table 12 - Web Service Infrastructure Deployment Stages

Considerations and Inhibitors

Several considerations need to be made when upgrading the infrastructure for Web Services, some of which will be potential inhibitors

- Project Culture

  Working against the implementation of an enterprise-wide Web Services infrastructure is the project-centric culture that predominates today. In our survey of organizations who have implemented Web Services, 70% of the projects were funded at the divisional level. As such many questions regarding the responsibility, funding or even the need for an enterprise-wide approach remain cultural rather than technical.
- Interoperability

  With Web Service protocols evolving rapidly their implementation across the diverse infrastructure is likely to raise issues of interoperability due to different versions and inconsistent implementations by vendors. Infrastructure elements will need to be upgraded in parallel to avoid interoperability issues.
- Lowest common denominator functionality

  Additionally, the diversity of infrastructure elements means that not all will deliver the same Web Services capabilities. E.g. certain emerging protocols will only be supported by some elements.
- Performance

  Web Service infrastructure will at least in the near term often constitute an additional layer of infrastructure, as opposed to replacement, impacting performance. However, this should only affect the Web Services themselves, not other messages and transactions passing through the same infrastructure element. Organizations can also consider introducing parallel infrastructure elements dedicated to Web Services to ensure existing operations are not affected.
- Service Level Agreements

  Consider how Web Service infrastructure elements contribute to, or impact, the delivery of SLAs. Consider what WSM capabilities might be needed to manage and monitor SLAs. What SLA is offered by the operational Web Services environment to the application/service developer?

Needs of the Large Enterprise

Large and global enterprises face a similar challenge of upgrading for Web Services as they do with any other infrastructure decision. E.g. the project culture issues raised earlier. Web Services are unlikely to change current practice without an associated change in culture. However, given that Web Services are primarily being introduced at present in large enterprises to support EAI needs, it once again raises the issue. We believe it is sensible for Central IT to deliver or at least provide guidelines on the following,

- Global Private UDDI Registry. There seems little point in allowing each division to implement their own
- Protocol Usage and Interoperability Standards. Guidelines as to which Web Service protocols can be used should be issued to enable enterprise wide interoperability and integration, and the use of WS-I profiles
- Reference Platform. Issue guidelines for recommend platform(s) that support above.
- Messaging Infrastructure. Upgrades to network to transport Web Services messages.
- Security – The needs for a common security policy should be apparent.

Roadmap Actions	
Plan & Manage	Identify infrastructure upgrade required for new technical strategy, in each stage Assess Organization impacts. E.g. Project vs central responsibilities Acquire funding for infrastructure upgrades Set SLA policies Establish WS security policy
Infrastructure	Adopt phased approach to infrastructure upgrades. Upgrade infrastructure in step with needs, and protocol evolution including: - WS Developer environment - Hosting environment for WS facades - Hosting environment for WS applications - On Demand Operating Environment - Support for Consumer environment - Developer tools - Security infrastructure - Monitoring and measurement - Diagnostics and failure etc
Architecture	Plan WSM deployment architecture, e.g. extent of distributed elements Establish SOA middleware layer
Process	Publish interoperability strategy and guidelines
Projects	Upgrades to centralised infrastructure Assess WSM requirements Private UDDI implementation

Moving to SOA

Service Oriented Architecture or SOA is the subject of the moment. It's a great idea, to publish a portfolio of services from the existing application base that can be easily reused by existing and new applications without invasive activity in the source application. But many enterprises are finding it's not as straightforward as slapping loose coupled interfaces onto existing applications and legacy code. Genuinely adaptable architectures need a little more thought. In this report we examine the issues in moving to SOA.

"To the extent that the components of business are off-the-shelf services, then the business attention switches to the configuration of the business process. If we are going to talk about components, whatever they are, we've also got to talk about the relationships and interactions and collaborations and connections between components, the "glue" or "wiring". In plugging together a component based business, we need to pay attention to the interfaces between the business components - in other words the business relationships."

Richard Veryard, The Component Based Business

Introduction

First it is useful to differentiate between Service Oriented Architecture (SOA) and Web Services. SOA is the broad set of concepts that enable units of functionality to be provided and consumed as Services. This essentially simple concept can and should be used, not just for Web Services, but also at each tier of the architecture, in order to compartmentalize and provide flexibility.

What some enlightened organizations have been doing over the past year or so is restructuring their application base to expose core services so that they can be reused in a loose coupled manner. By loose coupled we are not necessarily referring to specific protocols or behaviors, but the reduction of dependency and increased separation that allows the core service to be more easily used. And more easily means lower resource, lower cost and faster change.

Whilst WSXX based protocols will naturally be used for publishing Services, various technologies will be utilized in creating the Service Oriented architecture. The really far sighted organizations will be implementing WSDL and a UDDI compliant registry throughout their architecture in order to formalize the publication of Service meta data, and to make reuse easier.

For more on SOA see related CBDI reports[18].

[18] Services Oriented Architecture - A Series of CBDI Reports by Oliver Sims

This series is about the effective specification, design, and delivery of service-oriented applications and business processes in the enterprise environment. It assumes that applications must integrate not only with legacy, but also with each other, in order to avoid creating tomorrow's stovepipe legacy. In particular, we address the major "choke points" from end-to-end of the development lifecycle, and end-to-end from front to back of the distributed enterprise system.

Part 1 -The Foundation - http://www.cbdiforum.com/secure/interact/2003-03/foundation.php3
Part 2 - The Bridge - http://www.cbdiforum.com/secure/interact/2003-04/bridge.php3
Part 3 - Federation - http://www.cbdiforum.com/secure/interact/2003-05/federation.php3
Part 4 - The Platform - http://www.cbdiforum.com/secure/interact/2003-06/federation.php3
Part 5 - The Service Based Business - http://www.cbdiforum.com/secure/interact/2003-07/services_oriented_arch.php3

SOA Attribute	Description
Business Level Services	Services are published at a level of abstraction that corresponds to real world activity and recognizable business function. The really compelling aspect of this is the opportunity to implement comprehensive alignment and integration of the service life cycle with the business product and/or process life cycle.
Service Based Collaboration	Although Services are being widely used internally and for integration purposes this technical orientation will change soon enough. Services will increasingly mirror real world business activity, such that data is obtained from the real source in real time, that combinations of Services from collaborating organizations cooperate to provide value added Services. Whilst there will be infrastructural differences covering matters such as security between internal and externally supplied Services, increasingly there will be a common Service model that allows seamless operation of business processes internal and external to the enterprise. Although Services may be simple, they may also be aggregated from different sources, again reflecting real world business activities. There is clearly a real requirement for Service interaction and dependency modeling.
Separation of Interface from Implementation	A core tenet of SOA is that it's the interface (as opposed to the application) that is integrated, in a manner that the consumer has no visibility of the implementation. Some call this "Interface Based Design", a well understood technique from Component Based Development. However this is a rather technical perspective. Services are offered at a business level of abstraction, which renders the interface a business interface, and this generally means a contract, which is expressed in XML.
Contract based integration	The importance of the contract in SOA cannot be overestimated. The formal contract is the device that allows us to create virtual businesses, formalize system and scope boundaries, minimize dependencies and maximize adaptability, use black box testing and have a choice of Services and more easily change supplier. Although there is some work in progress to bring Design by Contract into UML, and a vague intent to formalize contract elements in BPEL, this looks like an area with a lot of outstanding issues.
Separation between Provider and Consumer	Among other things, Service Oriented Architectures must be designed with a view to the ease of management - including supply risk management. If the enterprise is dependent upon a few key service providers, this represents a potential risk to the enterprise. This leads to a design goal of making service specifications as general as possible, which of course brings us directly into conflict with performance objectives.

Table 13 - SOA Attributes

It's About Better Integration

Although Web Services have stolen the limelight, what's actually happening in private, in enlightened companies, behind their enterprise firewalls, is restructuring. Because the business case for restructuring on service lines is obvious - it avoids application rebuild, it reuses what already exists and allows IT organizations to respond much more rapidly to new and changing requirements. That's going to reduce costs.

Many organizations tried to reuse (extend the life of, implement best of breed components) their existing applications by using Enterprise Application Integration (EAI). However most EAI efforts failed to cut it because, the most widely adopted approach was to tightly couple the implementations. For example projects were focused typically focused on application specific objectives such as interfacing Oracle with Seibel, rather than the logical level of creating single instances of collaborating business services. In contrast the Service based architecture establishes a more durable Services layer, where the integration point is the Service specification or the interface, not the implementation. This provides implementation transparency, where multiple implementations may be rationalized, or an older implementation upgraded, with minimal impact on the consumer of the Service. This establishes a loosely coupled architecture of services that have minimum dependencies and maximum platform independence that can be reused with minimum cost overhead.

CASE STUDY - THE CHERNOBYL STRATEGY

In one situation we are familiar with, the application platform tool combination had just been announced as obsolete. Basic support will be continued but no further upgrade in function. You know how it works - the original tool vendor fell on hard times and was bought out by a specialist in legacy product management. In this case the applications were about ten years old, and as we all know, that's relatively young for an enterprise system. So what to do? There's almost never a case for rewriting the applications in some modern language. If the core transactions fit the business need, and the platform tool combination is adequate for basic server side transactions, and it usually is, then you may be able to use our Chernobyl strategy. In this situation you encase core transactions and possibly data in thick (logical) concrete, and expose core business transactions and data that can be aggregated, new rules applied, and potentially extended on a middle tier, before exposing them as business level services. Many organizations have followed something similar to this approach with their mainframe applications, although in practice the sharp focus on exposing Business Services hasn't always been present.

Creating the Foundation

In the first part of our report series on SOA (reference above) we advise on the importance of using service thinking inside the enterprise.

There is no reason why these should not apply inside enterprise IT as well as outside. The result would be a single kind of interface, using the same technology, for all internal systems that provide a service. Many existing systems would need to be wrapped of course. And performance would have to be considered. However, the potential advantage of a single interface type, that maps to many programming languages, is a huge simplifier for enterprise systems. It's like a common hub—indeed, considering a number of other technologies that go with web services—such as message queuing, it's almost a must—a highly compelling simplifier. And simplification means effort reduction which means cost reduction and/or faster response to business needs.

Note that this also reduces the variety of invocation mechanisms. Each of these has its own programming model, and this is often visible in the applications making the requests. Wrapping all of these mechanisms with web services not only provides simplicity for the application developers, it also separates the communications and messaging infrastructures from applications. This means they can be evolved without impact on applications.

In the report we then discuss the limitations of wrapping. Sometimes it's the only possible way forward, as with the Chernobyl strategy discussed in above. But as with physical buildings, the strongest foundations will be established by a good underlying architecture. And this will be component based.

Best practice OO design suggests that objects should provide a defined service. It's the same with components. We identify two kinds of component - process components, which make use of the services of entity components. These will ideally be arranged according to the mediator *pattern, which reduces dependencies. This makes the mediated components more re-usable by other process components. The component assembly is therefore complete in itself, and needs no code to glue the components together - the component middleware effectively does that dynamically at run-time. This is where the Generic Component Realization comes in. A Generic Component Realization is one where the component consists of a declarative specification or script, which is interpreted at run-time by some middleware engine - for example, a workflow engine. Such an engine can be thought of as a generic implementation, modified by the script, for all components that make use of that particular engine. This means that a workflow or B2B collaboration can be developed using many of the same concepts as components destined to be implemented on a J2EE or .NET environment.*

In summary, the component concept can be applied to a wide variety of modules, whether built with compiled code, or declaratively scripted. Further, the autonomy characteristic of all components, from design through build into the run-time, means that its internals are opaque to all but the builder. The service oriented architecture is a classic component based architecture, but complemented with many different types of implementation and a service invocation layer.

- *Provides, through use of web services interfaces, a single interface and service access design, which is independent of the underlying platforms*
- *Provides a single type system for interactions across and outside the enterprise*
- *Provides a clear architecture for the internal implementation of services*
- *Separates much more clearly the business logic in code, workflow, or B2B collaboration specifications from the underlying middleware, including integration subsystems, communication subsystems, and component containers*
- *Simplifies the whole enterprise development environment*

Foundation Characteristics

The Business Service Bus

In the days when middleware was top of the technical architect's toy box, the notion of the transaction bus was very popular. But whilst this is a necessary layer, it's equally if not more important to develop the concept of the Business Service Bus. We first introduced this notion over two years ago, and since then it seems to have gained widespread acceptance because it is inherently simple.

The Business Service Bus is the set of business services for a specific domain that are available for widespread use across an enterprise. The services are published in a UDDI compliant registry which allows them to be reused without manual intervention by the provider. Usage policies are implemented as part of an authentication and approval system, which differs from external services only in relation to being inside the firewall. The services have been implemented using a standardized semantic set that normalizes local application semantics and rules.

One of the reasons for using the Business Service Bus is so that common specifications, policies, etc can be made at the bus level, rather than for each individual service. For example, services on a bus should all follow the same semantic standards, adhere to the same security policy, all point to the same global model of the domain. It also facilitates the implementation of a number of common, lower-level business infrastructure services that can be aggregated into other higher level business services on the same bus (e.g. they all use the same product code validation service).

Many organizations have attempted with mixed success to implement common components by rationalizing their installed application base. In contrast the Business Service Bus is based on the premise that there will be multiple implementations of the same business object, either now or in the future, and the purpose of the bus is to make those implementations transparent from service usage.

The overall objective is by definition, to establish a single logical bus structure. Whilst it might be natural (and politically easier) to implement bus structures that mirror organizational boundaries, the successful organization will see the real value in creating cross organizational services that allow the organization to evolve independently of the information services.

A Framework for SOA

- *Provides business level visibility of available and planned services*
- *In an organizational independent manner*
- *Provides a clustering mechanism for managing service attributes including semantics, taxonomies, usage, security policies, SLAs, funding and charging models, etc*
- *Bridges business and IT perspectives in a precise and meaningful manner*

Business Service Bus

A service oriented architecture is one view or perspective of the overall architecture. It's a mechanism for ensuring the application and infrastructure is and remains loosely coupled. Of course the SOA is implemented as a series of alterations or delta to conventional practice, and we need to determine for any particular enterprise what that delta is.

In Table 14 we provide a framework, as a useful way to document this, and both ensure some consistency and completeness of the list. We have commenced with something like a Zachman framework, which is widely used by enterprise architects, and then developed a fairly arbitrary set of domains, against which we can record SOA decisions and or deliverables. The Zachman framework analyses architectural elements across conceptual, logical and physical layers, but we have found it more useful to think about requirements, specifications and implementations.

However we do stress this is a framework in the best meaning of the term, and encourage organizations to use this as a starting point, and to develop and customize as appropriate. The advantage of this framework level of abstraction is that it moves just a little beyond vague, and ill understood terms like "architecture", while retaining the ability to have an overall perspective. A framework of this nature would also form a good communication vehicle between the (architecture) provider and consumer. It also has the advantage of allowing specialist architects to develop customized views that can easily be mapped to other views.

Another important aspect of SOA is the question of how to ensure architectural decisions get implemented - otherwise known as the governance issue. Each deliverable can be attributed with governance roles, including standards or guidelines, mandatory or optional, which themselves have applicability to specific domains which might include platform, product, layer, application, relationship etc. We provide some examples of governance deliverables in Table 15

Domain / View	Requirements (Conceptual)	Specifications (Logical)	Implementations (Physical)
Business	Standardized business services	Business Service Bus	WSDL specifications
	Business virtualization	BPO Service architecture	UDDI based service registry
	Purely information based products	Information service specifications	WSDL specifications
	Service based product components providing value add to physical products	Service product specifications	WSDL specifications
Information	Definition of the real information owner, internal and external to the business and strategy for managing that class of information	Information currency and ownership distribution analysis	Information access services
	Common semantics and domain applicability (business unit, enterprise, ecosystem, sector etc)	Semantic standards	XML documents and mapping and transformation rules
Application		Applications redefined as sets of business objects and related services	Applications acquired as (hosted) services
Service	Service is first order construct in business process	Service as unit of provision and consumption	Delivered service
	Standard/common business services	Standard/common business services	Standard/common business services
		Standard/common infrastructural services	Standard/common infrastructural services
	Business service patterns	Domain business service patterns	
	Business service contract templates	Service specification contract templates and reusable components	BPEL4WS or ebXML template specifications
Component	Business domains mirror business requirements for articulation	Business components encapsulate a single business concept (entity or process)	Offers well defined network interfaces that offer services
Middleware	Business Service Bus	Wire protocol standards	XML Messages
		Message security services	Message security services

Domain / View	Requirements (Conceptual)	Specifications (Logical)	Implementations (Physical)
			Component and Service container
Service Management	Business rules	Rule specifications	XSLT or similar
	Business policies	Policy specifications	XSLT or similar
	Business service requirements	Service Level Agreements	Dependency specifications, thresholds, contingency plans
	Business trust requirement	Trust specification	Monitoring policies and rules, breakpoints and escalation specifications
		Management services pipeline	Management services pipeline
Platform		Grid based services virtualize physical platform resource	Platform functionality delivered as services
		Integrity Units define domains for trust, resource management, technical upgrade	Virtual physical resources
Device	Device independent UI	Service interface specification NOT the User Interface	Service interface specification NOT the User Interface
User	Roles	User profiles	Directory Server

Table 14 - Service Oriented Architecture Framework

Architecture Deliverable Type	Governance Role		Domain applicability
	Standard or guideline	Mandatory or optional	
Patterns			
Templates			
Common components			
Common services			
Protocols			
Semantics			
Products			
Practices			

Table 15 - Architectural Governance

Summary

Most organizations should now be planning and executing on some level of SOA based environment. For some the change will happen by default as they implement new versions of packages such as SAP that have embraced the concepts. But most organizations have a heterogeneous environment and need to manage the transition to ensure they achieve a high level of componentization and separation, that will flow through into improved economics and response time to change.

The archetypal enterprise organization is highly distributed, but in a service context it is very important that there is coordination of service creation and reuse, to ensure the common usage where necessary. Governance policies are an essential pre-requisite to make this happen. It won't happen without serious cross organizational effort.

Roadmap Actions
Define a Business Service Bus Establish a vehicle that enables policy development and communications at the service level between IT and business communities.
Develop a component based architecture to support the Business Service Bus Make plans on how clear separation will be established at the (application) implementation level. Build into all project and acquisition plans. Ensure that acquisitions confirm to separation policies
Implement a Service Based Scoping Policy for Projects Ensure that all projects are required to scope and justify their activity on the basis of services used and implemented.
Implement Relevant Governance Mechanisms Implement appropriate practices to ensure that corporate SOA strategy gets implemented in delivered and acquired applications.

ISVs and Packaged Application Vendors Start Here

This section considers how Web Services will impact ISVs and how they must adapt their software packages to support both the provision and consumption of Web Services.

Introduction

Independent Software Vendors (ISV) must also make the transition to Web Services. As with any major change, Web Services presents the ISVs both with opportunities for new business, and challenges in making the transition. Some considerations ISVs should make include

- Near term, customers will expect to use their existing packaged applications in Web Service Scenarios and ISVs will need to consider to what extent they need to expose Web Services directly from their application to enable this.
- Longer term, ISV's must look to see how their products can add real value and deliver ROI to their customers by being WS enabled. Customers can be expected to consider new packaged applications that exploit Web Services, and ISVs should evaluate the opportunities this presents them.
- ISVs need to consider how their products are going to participate in more dynamic, collaborative business processes that will be supported by emerging Web Service protocols.
- The impact on the ISV's business model. E.g. changing from software to service provider. And whether Web Services introduce new competitive threats from new forms of competitors
- What infrastructure ISVs should use to deliver their Web Services

How Web Services Will Impact the ISV
Opportunities and Benefits
Will ease customer's integration requirements
Providing SOAP/WSDL interfaces may be straightforward
Introduce new opportunities for collaborative applications
Change deployment options. E.g. Bring new 'service hosting' opportunities
Re-engineering Challenges
Highlight weaknesses in, or unsuitability of, current interfaces
Demand for more real-time behaviour in their apps
Need for apps to consume Web Services, not just provide them
Converting apps to support emerging enterprise-level Web Service standards for transactions and process orchestration
Remove internal dependencies so that customers can use services at a more granular level than the whole application

Many package vendors for example are converting interfaces and technologies initially developed to address Enteprise Application Integration (EAI) requirements, to support Web Service Protocols. Adapting an existing interface to support basic Web Service protocols such as SOAP and WSDL should not be too difficult. In many cases this might be achieved through a simple 'wrapper' that requires little modification of the base software. Elements of WS-Security can probably be addressed in this manner too.

However, the more complex protocols that are emerging to support dynamic, collaborative business processes, or that might be considered pre-requisites to 'enterprise level' Web Services, such as transactions, and business process orchestration, will likely require some level of re-engineering of the software.

Additionally, ISVs must recognise that not only must their software expose Web Services, but will often need to be re-engineered to consume them too.

ISV Deltas

In several respects, Web Services will have similar impacts on the ISV as they do end-user organizations. Activities such as moving to SOA, considering how to best transition their applications to Web Services, and effecting that change, will contain many of the same actions for both ISV and end-user alike. However, the nature of ISV applications is such that there will be deltas in comparison to end-user adoption of Web Services. The prime difference of course is that Web Services may have not just an architectural impact on the ISVs applications, but also a commercial impact on their business model, on the way in which they deploy software to their customers, and how they charge for it.

Area	Roadmap Consideration
Business Integration Today, incompatibilities are resolved by EAI tools. Web Services will remove the need for protocol conversion, but not fully resolve semantic differences. ISVs can reduce their dependence on EAI-like solutions by conforming to standardised business semantics	Consider participation in industry-body activity setting Web Service standards for your domain Work with obvious ISV partners in eco-system to develop common business semantics
Collaborative Solutions Increased demand from customers to build collaborative solutions will mean few ISV applications can 'stand alone'. Applications must not just be able to share information via Web Services, but collaborate in common business processes	Move to SOA Ensure proper separation of concerns, so that business process is properly isolated Prepare for shift to standardised workflow and BPO – e.g. BPEL4WS, WS-Choreography
Horizontal Applications Applications not specific to a vertical domain will find themselves the most obvious candidates for deployment as hosted services, and affected by shifts to On-Demand and Grid based approaches	Ensure applications are 'location' independent Applications need to be properly componentised to exploit rapid, on -demand deployment Internal dependencies need to use Web Services to facilitate distributed deployment
Platform Independent Solutions ISVs often base their applications on a platform independent infrastructure they have built or acquired to enable portability. How will this transition to Web Services?	ISVs need to careful evaluate the use of platform specific Web Services infrastructure with regard to delivering portable solutions Implementation of SOA principals, a proper service architecture, use of the service bus, and a separated services layer will improve the ability to deliver platform independent solutions, whilst at the same time exploiting platform specific features
Configuration and Deployment Web Services enable widely distributed applications to be more easily integrated via services. Externally hosted applications can be integrated to the same level as internal ones	ISVs need to whether they can take advantage of alternative deployment patterns such as providing hosted services.

Table 16 - ISV Considerations

ISV Deployment Patterns

We see the following three basic patterns to the application of Web Services by ISVs

1. Traditional Deployment plus Web Services

The ISV delivers the software to the customer who hosts the application in-house.

Web Services are added to the application by the ISV for two reasons

- Internal Services. So that the customer can integrate the application more easily with other internal applications
- External Services. So that the customer can expose the services to their own external customers and business partners

2. ASP with Web Services - Hosted Service Provider

The ISV hosts the software themselves, or uses a 3rd party host. The customer then uses the application via its external Web Services.

Most ISVs do not traditionally host their own applications preferring instead (or more commonly the customer preferring) to use ASPs if the customer does not want to deploy the application in-house, and there is little reason to believe this would change.

Whereas traditional ASP was primarily a 'self service' approach with the application being accessed by employees via a browser interface, the use of Web Services enables the customer to integrate the hosted application into their business processes as if it was deployed in-house.

3. Distributed

A hybrid approach is a distributed pattern in which the customer still deploys part of the application in-house, whilst the ISV (or their ASP) delivers the remainder as a hosted service. Web Services could be leveraged to simplify the deployment and configuration requirement in situations where the ISV is responsible for 'feeding' information or rules that must be maintained on a regular basis.

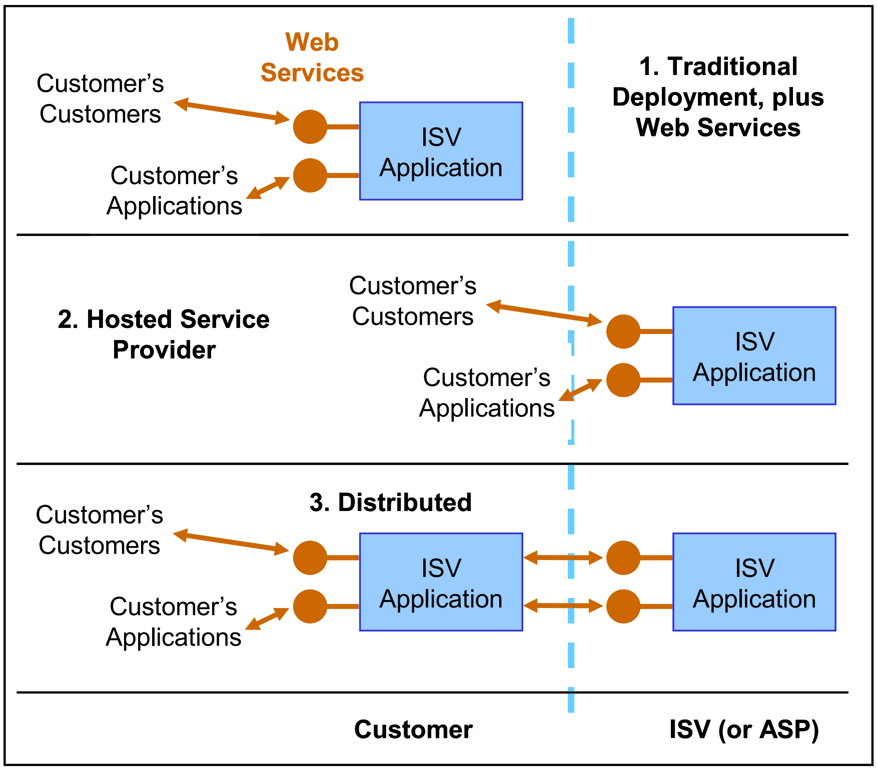


Figure 10 - ISV Web Service Patterns

Consuming Web Services

Regardless of the deployment pattern, ISVs will also need to adapt their applications to consume services, not just provide them. This could be for either

- Business Services – the application will need to consume Web Services provided by the customer's business partners and their own applications.
- Infrastructure Services – the ISV can leverage infrastructure services rather than having to implement certain capabilities within their software. Similarly, their customer will want integrate the software with a variety of infrastructure services provided for example by the underlying platform, Web Service Management products, and external hosted services.

Consuming Web Services will be the more challenging for ISVs. Though they may not be the ideal services, existing interfaces already exposed by their product can be quickly converted to use Web Service protocols. This may even be provided by a wrapper layer that removes the need to change the existing software. However, if existing ISV software is to consume external Web Services it will likely require some re-engineering, which may be significant.

Competitive Threats

Many ISVs will find themselves driven to Web Services as a competitive response. Agile ISVs looking for competitive advantage will use Web Services sooner rather than later. Though initial usage may well be only skin deep, it will quickly become a hygiene factor that ISVs must adopt to keep up. Long term, ISVs will need to re-engineer their applications to exploit Web Services. Thought this will be more expensive and difficult, ISVs who move fully to SOA to support this will undoubtedly gain long term competitive advantage in terms of agility and flexibility over those that simply try to dress up their existing interfaces with new protocols.

Web Service Strategies of Some Leading Package Vendors

SAP – SAP Exchange Infrastructure
http://www.sap.com/solutions/netweaver/keycapabilities/xi.asp

Peoplesoft –Pure Internet Architecture
http://www.peoplesoft.com/corp/en/products/technology/index.asp

IFS - Web Services and Integration
http://www.ifsworld.com/ifs_applications/technology/web_services_integration.asp#1

The business models of some ISVs will be threatened by increased use of Hosted Services that will introduce new types of competitors. ISVs have for some time been able to adopt the ASP approach. Web Services improves the integration and granularity of the hosted solution. However, it also introduces the possibility that end-user organizations could open up their own applications and provided them as hosted services in certain situations. Perhaps the best example today is Amazon[19] who enable other retailers to use Amazon's systems as a basis for their own e-commerce operation, which can now be better integrated with the retailers systems through the use of Web Services. This effectively places Amazon in competition with ISVs selling e-commerce solutions.

Web Services Adoption Steps

ISVs can consider the following outline steps towards adopting Web Services in their products

1. Wrap existing interfaces with SOAP and WSDL to provide basic Web Services support. Some of this ability will come courtesy of the application container the software runs in (such as J2EE, .NET, etc), and require minimal effort.
2. Building a more comprehensive Web Services Façade can be useful to deliver more meaningful aggregate business services, rather than simply exposing the existing APIs, and can be used to provide support for emerging enterprise level Web Service protocols that may require some application specific behaviour.

[19] www.amazon.com/webservices

3. Perform invasive surgery on the application to enable it to consume relevant Web Services rather than simply provide them. This probably requires the use of a "plug point" approach as the specific Web Services may be unknown in advance.
4. Start to use the Web Services you provide for end customers to integrate your own software components (eat your own dog food). This will prepare for more radical re-engineering in future – such as the move to an on demand operating environment
5. Consider what Web Service infrastructure elements you might acquire, licence or partner with vendors to provide, that would benefit customers in deploying or accessing your applications as Web Services. For example including components of Web Service Management.
6. Consider how a hosted version of the software would be deployed, what Services would be exposed, and what infrastructure would be necessary to manage users, usage, and SLA.
7. Re-architect where relevant to support a truly distributed implementation where individual components could be redeployed on demand

Roadmap Actions		
	Near Term	**Mid Term**
Plan & Manage	Agree roadmap - overall plan for transitioning Identify business opportunity/competitive threats	Consider new competitive threats introduced by hosted services, and non-traditional vendors (e.g. Amazon) Revised pricing strategy
Infrastructure	Deployment to Web Service enabled platforms Implement WS based security and trust	Consider build vs buy vs service usage equation for some infrastructure elements Incorporate hosting requirements
Architecture	Define route to SOA and release plans Publish new technical directions and architecture Wrap existing interfaces with base Web Service protocols, publish WS based API structure. Built a Web Service Façade Collaborate on WS based semantics within customer, industry or ecosystem groupings	Re-engineer to exploit "Enterprise" Web Service protocols, and on demand operating environments.
Process	Provide customers with upgrade path	Identify opportunities to collaborate with other ISV products real time access to information - real time access to new complementary functionality
Projects	Determine release contents and road to SOA	Redefine product(s) as sets of services Potential for hosted services

Applying Web Services

In this section we provide an analysis of the types of Web Service application, with a framework for planning when, where and why they may be applied.

Introduction

Eventually most software functionality will be published and consumed as Web Services. Three years ago this statement would have met with considerable scepticism, today many will agree. The entire industry is focused on this direction and, whilst there will be continuous evolution and morphing of the concepts, the momentum is now unstoppable.

However it will take some considerable time before this happens. First the standards, practices, tools and platform infrastructure are immature, second enterprise and ISV organizations have a both a huge portfolio of existing applications which need to be upgraded and a significant infrastructure and skills and practice development task. So meantime organizations need to a direct their energies in a manner that a) minimizes risk, b) enables learning in a controlled manner and c) maximizes the business impact of their early learning activity. In this report we provide a framework to assist organizations in making these critical decisions.

Conversion Scenarios – The First Steps towards Web Services

Existing Scenario	Conversion Scenarios	Drivers
Distributed Computing with Web Services	Integrating Components of disparate technologies such as .NET and J2EE. using Internet protocols	Platform independence Enable Wide Area Web-based distributed computing Technical simplicity
Web Site to Web Service	Converting functionality exposed through existing web site	Enable automation rather than self-service through browser Extend reach by enabling embedding in 3rd party sites
Portal	Using Web Services to integrate back end systems into portal Delivering functionality that was encapsulated in portlet as a Web Service to enable integration	Automation rather than self service Greater flexibility in portal publishing and integration Expose functionality to external portals
EAI with Web Services	Standards-based Web Service wrappers to replace adaptors Using EAI to wrap existing systems to expose Web Services	Reduce dependence on proprietary EAI adaptors (standards based integration) Leverage existing systems Faster, cheaper integration
EDI/B2B	Offering External Web Services across supply chain	Reduce dependence on proprietary EDI/B2B software Enable small partners to participate without expensive EDI software

Table 17 - Web Service "Conversion" Scenarios

Originally, Web Services were primarily presented as a way to expose object/component interfaces in such a way that they could be accessed across the Internet and outside the firewall – hence “Object Access” in SOAP. However, as SOAP went through the standards process the Web Service concept morphed into something more broadly applicable. Particularly important is the realization that the implementation behind a Web Service need not necessarily be object/component based and that it is equally valid for a Web Service to simply exchange documents as well as call methods.

Table 17 illustrates the wide variation of scenarios and styles that Web Services will be used in, all using the same protocols. Looking at many of the applications of Web Services so far it is clear their usage has been in what we term “Conversion Scenarios”, as listed in Table 17. Organizations are typically converting or extending some existing business process and technology scenario to use Web Services, and whilst this has brought many benefits they have not yet exploited the new capabilities offered by Web Services to re-engineer their processes. For example there is little use so far of the more dynamic capabilities enabled by Web Services such as publishing, locating and consuming services at run-time.

It's not unusual that new technologies are initially deployed as an incremental or alternative function. No surprise therefore that Web Services often used in parallel to the existing business process and technology to provide an alternative access to information.

The challenge for organizations today is that each of these existing scenarios is normally associated with its own set of proprietary technologies. That is, the technology used for EDI is not suitable for distributed computing, which in turn not suitable for EAI, etc. As a consequence we have 5 times the products, 5 times the cost, training and skills etc

This can becomes a problem as organizations extend integration with new business processes that span not only their own organization, but also their partners and customers, then none of these existing solutions is really adequate. When customers, partners and internal systems are all involved in the same end-to-end process then organizations need to question the following

- Where for example, does EAI end and B2B Start?
- Is there a break in the process between them? Is the end-to-end process broken by manual steps, or batch transfers that introduce delays and errors?
- Does the need for the Real Time Enterprise or Straight Through Processing stop at your organizational boundary?
- How is the accuracy and visibility of information improved if the real source is external? Is the information you supply your customers and partners accurate and visible in real time?
- How does your partner provide information to your customer? Directly or via you?

As the scope of integration grows organizations need to use aspects of each of the existing scenario for the complete solution. But today because of their proprietary nature is can be very complex to string them all together. A key advantage that Web Services brings is that you can start to adopt a single standard framework that can be applied to all of these existing scenarios rather than continuing to use point solutions.

Ultimately, the use of Web Services blurs the distinction between the existing scenarios but as they are often the starting point for Web Service projects the following section examines in more detail some of the reasons for using Web Services in them.

Distributed Computing with Web Services

Web Services are simply a vastly improved form of distributed computing. Whilst some organizations have been successful with distributed computing deployments, the limitations are obvious.

Exposing existing or new component interfaces as SOAP has been made very easy by the various platform vendors, and is the most common application of Web Services to date. Web Services protocols are now effectively embedded within the platform making their usage transparent to the developer.

Advantages

- Technology independent
- Protocol Based
- Open Standards

- Loosely coupled
- Works inside and outside across the firewall
- Richer Specification

Suitability

- Exposing business algorithms, not just exposing information
- Component/object based systems
- Potentially complex transactions with emerging Web Service protocols
- Remote Procedure Call behavior. Though current best practice is now moving towards recommending the asynchronous document exchange style of Web Service for many implementations.

Web Site to Web Service

Converting existing website functionality to enable application to application connectivity through Web Services is one of the easiest of conversions to perform. There is no reason why existing e-commerce applications cannot be converted to Web Services either.

Advantages

- Leverage investments already made in web enabling existing systems
- Convert the existing information, forms etc., presented through web browser for customer, employee and partner self-service
- Use existing technology infrastructure, with minimal amount of effort required to upgrade to support Web Services
- Web Services directly supported, and relatively easy to implement in current Web Servers such as Apache, WebSphere, Windows Server, Weblogic, etc
- Technology used to provide secure external Web Sites can be used to used to provide external, secure Web Services

Suitability

- Information provision and Simple transactions. Even with emerging standards it is probably that existing form based website functionality has been designed in a way to support more complex transaction scenarios.
- Simple e-commerce transactions. For example providing Web Services to enable business partner to embed them in their own portal and resell products, but have the transaction made directly with the provider
- You don't expect communications to be any more reliable or robust than to your current web site

Portal with Web Services

The use of Web Services to extend portal usage is equally useful to the service consumer and provider, who

- Consume Web Services to build the portal. This doesn't really differ from consuming Web Services in any other scenario, except that portals by their very nature tend to consolidate information from diverse sources which makes Web Services ideal as a mechanism to provide that information.
- Provide Web Services by converting existing Portlets to enable remote consumption
- Wider use of Web Services within portal scenarios will increase as emerging standards (WSRP[20]) are adopted

Advantages

- Portal Technology independence. Portals are often built using proprietary portal engines.

[20] Web Service Remote Portlets http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

- Ease of integration for remote portal builder
- Portal vendors now providing Web Service support

Suitability

- As with web site

EAI with Web Services

Limitations of conventional EAI can be addressed to a great extent by Web Services. By converting the Enterprise Applications to provide Web Services natively for their interfaces, you can eliminate most requirements for adaptors. Many of the Packaged Applications that are the centre of EAI scenarios, for example SAP, have already been converted by their vendors to expose Web Services. However EAI doesn't disappear completely.

- Web Services may not an immediate solution for all legacy applications as end-user organizations may not have the time or resources to build wrappers around them. Conventional EAI can continue to provide the adaptors that convert them to Web Services. The platform vendor that underpins the legacy application may also provide such a solution.
- Adaptors will still be needed to now wrap existing applications with Web Services, where they are not provided natively
- The EAI hub typically provides other functionality that is still required for integration – most notably Business Process Orchestration. Though this is becoming Web Service based too. The challenge for EAI vendors will be to make their BPO engine Web Service compliant.
- Semantic conversion. Even though XML based, semantic conversion may still be necessary to convert documents passed between service consumer and provider unless they adhere to the same semantic standards – many of which are only just emerging.

So EAI doesn't go away – it just adapts to Web Services. Web Services should make it easier and faster to deliver EAI solutions. And perhaps most importantly, provide an EAI solution that is based on Open Standards rather than proprietary technology.

Advantages

- Standards based
- Remove some need for proprietary adaptors
- Leverage existing EAI infrastructure providing vendor provides Web Service upgrades
- Web Services can continue to use existing messaging infrastructure that may be part of the EAI architecture

Suitability

- Internal Web Services
- Exposing Web Services from Legacy systems
- Document exchange style of Web Services

EDI/B2B with Web Services

EDI and B2B is one of the most commonly presented Web Services scenarios. Today proprietary EDI and B2B gateways are widely used by large organizations but the high cost of implementation has prohibited their many smaller suppliers and partners from utilizing this approach. Frequently EDI is batch type of operation, and is not suitable to providing more real time information feeds, such as accurate stock availability, that may be required.

Thanks to the work of the EDI community, document exchange standards already exist and these are being evolved into XML and Web Service formats by initiatives such as XML.org. Additionally, ebXML.org has done much work to provide a XML-based standard protocol stack that includes SOAP messaging and a process orchestration mechanism that is needed to support standardized business processes to complement the document standards. However, much of the ebXML stack is in conflict with various similar Web Service initiatives. See the section on Web Service Protocols for more information.

Until the Web Service protocol stack matures, organizations can use ebXML for a comprehensive EDI/B2B solution though clearly there is the uncertainty of how these initiatives will merge in future.

Alternatively, organizations with simple EDI/B2B requirements can use the web site to Web Service or distributed computing conversion scenarios above to provide alternative lower cost access mechanisms for their partners, and deliver more timely information.

Advantages

- Lower cost of implementation, hence effectively available to all participants regardless of size
- Open standards rather than proprietary gateways and networks
- Enable real time exchanges

Suitability

- Near to middle term, complement existing EDI scenarios with Web Services for smaller participants

Where to Apply Web Services

Whilst we expect to see a gradual adoption of Web Services across all connectivity scenarios, we fully expect organizations to adopt a "if it ain't broke, don't fix it" attitude, and extension not replacement will be the de facto strategy.

In the short to medium term organizations will be looking for the sweet spots in which to use Web Services, where there are clear advantages or where Web Services provide a solution not really viable with the existing technologies. In this section we provide a framework for identifying these sweet spots.

Business Characteristics

Table 18 examines some of the characteristics of business processes that might be best supported by the applying Web Services.

Business Characteristic		Reasons to Apply Web Services
High number of participants	• One to Many • Many to many • Multiple intermediaries (business and infrastructure) • Federated	• Increasing likelihood of diverse technology across participants, and/or unknown technology at other end of wire • Support for Federation
Cross Functional	• Process or information need spans many business units, organizations	• Diverse technology
External integration	• Need to integrate customer or partners	• Increasing likelihood of incompatible technology between provider and consumer and likelihood that technology at other end of wire is unknown

Business Characteristic		Reasons to Apply Web Services
Real time information need	• Provide real time access to remote information • Improve the accuracy and timeliness of information • Retrieve information on demand from source rather than replicate	• Enable direct access to core operational systems, rather than cached or replicated data which is out of synch
Automation	• Application to Application • Repetitive, well defined processes, rules and information	• Provide automatable interface, rather than self-service browser interface • Human intervention for exceptions, not for every execution
Dynamic Provider/Consumer marketplace	• Dynamic selection of new service providers • Self subscription of service consumers	• Richer specification of Service • Programmable specification enables automation of service consumption in applications
Dynamic Process	• Reconfigure business process on demand	• Enable runtime selection of service and/or provider

Table 18 – Business Processes Characteristics suitable for Web Services

Looking for Boundaries

Boundary	Use of Web Services
Organization	External interfaces to customer, partner, etc use SOAP for loose coupling to remove technology independence
Division/Business Unit	Internal interfaces to other business units use SOAP. Loose coupling supports potential for outsourcing, spin off, etc as Organizational boundary
Sub Assembly (grouping of related objects and components)	(e.g. Customer) Internally uses native platform protocols to communicate between low level interfaces, but interfaces external to the sub assembly are SOAP to facilitate integration into multiple systems. Enable sub assembly level replacement or upgrade Enable outsourcing of business functionality at sub assembly level (e.g. logistics)
Application	As sub assembly. Enable Application level replacement or upgrade
Application Layer	Communications across layers such as presentation, business rules and data, or client/server use SOAP, but use platform native protocols within the same layer
Platform	Communication between components on same platform use native protocols, whereas communications to different platforms use SOAP

Table 19 - Candidate Boundaries for Web Service Adoption

Another useful selection process is to identify where processes have to cross what we term "boundaries". Where change might occur independently on either side of a boundary, then a loosely coupled approach (as offered by Web Services) to any connections across those boundaries can better facilitate those changes. Whereas within the boundary existing proprietary or platform specific technologies and approaches can continue to be used, and may be desirable for performance or SLA reasons.

An organizational boundary is an obvious candidate as the participants on either side of the boundary need not be forced to adopt the same technology to interoperate. But this can true for divisions and business units within the organization too. As well as organizational, other obvious boundaries are technology and application architecture as illustrated in Table 19.

In time this concept may become redundant as all communications simply become based on Web Services. In the meantime, identifying boundary points across which Web Services are most suitable is a useful starting point for Web Services adoption.

Exploitation Scenarios – Capitalizing on Web Services

Beyond the conversion scenarios outlined above, there are three exploitation scenarios based on re-engineering business processes:

1. **Reengineer Selected Information Flows.**
   Using the more conventional approaches outlined above, but reengineering the way information flows through the business process.
2. **Use Breakthrough Technology**
   Taking advantage of new capabilities offered by Web Services, particularly their support for more dynamic usage scenarios via
3. **Convergent Technology Strategy**
   Implementing Web Services in conjunction with other emerging technologies (which are themselves becoming increasingly Web Service based) such as
    a. Pervasive Computing
    b. Grid Computing
    c. “On Demand”, or “Utility” Computing
    d. Autonomic Computing

Or of course some combination of each.

Re-engineering Selected Information Flows

Web Services can be used to optimize and change what we term the Information Supply Chain – i.e. we don’t change the physical participants in the supply chain, but we do change how information flows between them. The following shows a couple of examples of how we might consider re-engineering firstly data sharing, and secondly process execution to improve information flows.

Data Sharing

With the capability of semi real-time, wide area interoperability, Web Services have the potential to enable breakthrough solutions. Some of the most interesting early case studies of Web Service adoption have been about radical improvement in data availability.

Today, data is commonly replicated by a plethora of diverse mechanisms amongst the participants in business processes. A conversion scenario might use Web Services to enhance the replication process.

Whilst this would simplify and standardize the way in which data is transferred, the use of Web Services per se doesn’t fully remove synchronization problems, nor does it necessarily improve the accuracy of information presented to a particular participant unless other participants replicate changes the moment they occur.

Figure 11 illustrates how participants might share the same single source of information using a Web Service so that all information is up to date. This is not to suggest that all participants now consolidate all their data in a single shared database. Rather it is the principle of each participant sharing the data they truly own and who’s responsibility it is to ensure that data is timely and accurate.

Advantages

- Timely and accurate information
- Reduction on in-house data storage requirements

Suitability

- Fast changing data where changes should be made available ASAP to other participants
- Large volumes of reference data that would otherwise need to be stored in-house

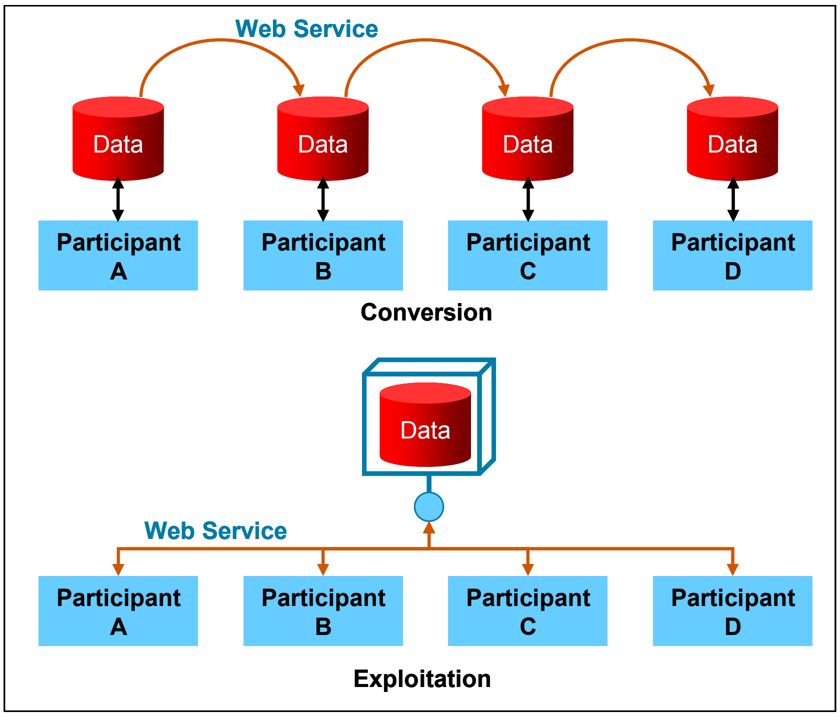


Figure 11 - Data Replication or Data Sharing?

Shared Collaborative Processes

The next logical step is to ask if participants are going to share the same data via Web Services, should they also consider sharing processes too? Today, not only will the participants be replicating data, but will often use that information in similar systems. In an exploitation scenario, rather than duplicate functionality in-house, organizations could share their processes as well as data. This is not the same pattern as Application Solution Provider (ASP), as the process is shared between all participants, rather than each of them just outsourcing their own. However, it may well be that a 3rd party such as an ASP would be ideal for hosting common shared processes as Web Services.

The BPEL protocol is suitable for this scenario.

Advantages

- Process and business rule consistency
- Reduction of in-house processing requirements
- Reduction of in-house configuration management

Suitability

- Common processes that are not the source of competitive advantage

Of course in both the above cases, whilst Web Services might provide a solution to current problems they introduce new challenges of their own. Common objections to the above scenarios would be security, SLA, scalability, and the risk of basing core systems on external dependencies. Whilst these must be weighed against the business benefits, we would also observe that apart from the additional communications factor that would be introduced, the same objections are also unfortunately often true of in-house systems, and the current replication process. For small organizations in particular, an external entity may be able to offer a more secure, scalable, robust solution than they could afford to operate in-house, enabling them to focus resources on the core processes and data which are unique to them, and they in turn will be offering to others as services.

Additionally it clearly requires some level of process and semantic standardization amongst participants, though replication requires this to some degree already. Moreover, where the participants are involved in many-to-many relationships with each other it requires industry wide agreement. See *"managing participation"* later in this document.

Re-engineering like this is likely to occur slowly, and probably take place between a limited number of very close partners before spreading out across all participants. In some industries, dominant participants might be able to force the change if they think it desirable. That said, there are already some examples of organizations providing Web Services as an alternative way to access information they own and which they normally deliver via replication.

Use Breakthrough Technology - Dynamic Web Service Consumption

One of the main advantages that Web Services offer over most existing application connectivity mechanisms is that consumption of Web Services can be resolved more easily at run time, and not just at design time. Two key elements of this are

- Rich specification, or "Self Describing" interfaces, which can be accessed programmatically to understand what a Web Service does and how it should be used
- Publication and Discovering mechanisms to announce and locate required Web Services

These elements can be built into applications themselves so that they can effectively discover and use new Web Services at runtime. However, most organizations are currently skeptical of any notion that Service consumers might appear to randomly and anonymously discover and use the provider's Services. They point out that in most cases, the necessary business relationship has to be in place long before any use of Web Services is appropriate, and as such any need to discover and consume the provider's Services at runtime is a moot point.

This is a valid point. However, in certain industries some common standard business transactions may well be suitable for this scenario. For example in the auto insurance industry where most insurance companies already provide a near universally similar "service" for insurance quotes via their web sites or call centres (i.e. some minor variation on give me the vehicle ID, your ID, and your location, and I will give you a quote) could easily be adapted to a Web Service scenario whereby my personal finance application running on my PC could each year automatically include new insurance companies in the list from which to get quotations. Though whether insurance companies really want to enable such an easy form of price comparison is of course a different question.

Advantages

- Remove developer effort to consume new Web Services
- Flexibility of choice in Service provider, resolved at run time

Suitability

- Common, standardized business services

Convergent Technology Strategy - Autonomic, Grid and On Demand Computing

An alternative use of the dynamic Web Service scenario outlined above is to support Autonomic, Grid and On Demand Computing. Rather than use the dynamic capabilities to discover new service providers, instead they can also be used to discover and consumer new services from a known provider – and of course internally.

These might be used for

- Failover and backup. Publishing, discovering and using alternative routes to, or instances of a Web Service in the event of failure. i.e. Self healing systems
- Scalability. Publishing, discovering and using additional instances of a Web Service to scale up at periods of high demand
- Versioning. Publishing, discovering and using new versions of a Web Service
- Location Transparency. Moving the implementation of a Web Service without impacting the service consumer

Given the current business reticence to use the dynamic Web Service consumption scenarios mentioned previously, it is probably that in the medium term much wider use of the dynamic nature of Web Services will be seen in the Autonomic, Grid and On Demand arena.

A more detailed look at the use of Web Services in the context of this is provided in the section *"Web Services Roadmap for the On Demand Business"*

Advantages

- Reduce developer effort, and time to solution via self healing, self discovering systems
- Reduce Web Service configuration management

Suitability

- Failover. Scalability, etc are required for all mission critical Web Services

Application Policy

In the Web Services Maturity Model we introduced the concept of Phases, which illustrate limitations in and the progressive approach to capability development. The phases and capabilities provide a framework for policy and decision making. For example, an organization might take the decision that shared data strategies are first order opportunities in the next two years, but that collaborative processes should only be used in non critical environments until certain standards and associated functionality are available.

These phases are not quantified in terms of time, but in terms of capability at any point in time. Let's look at these in terms of a policy framework, which will provide overriding guidance to project managers.

Phase	Status	Application policy guidance
Phase 1 - Early Learning	Little formalization or coordination; technical matters drive activity;	Internal or trusted users; exploratory; minimize investment; mostly non secure or using transport level security; non critical business function; low volume application
Phase 2 - Integration	Architectures established; some governance policies in place; some infrastructure and management in place; rudimentary SLA's established	Mostly internal usage; limited external users and or trusted partners;
Phase 3 Reengineering	Service based process engineering capability in place; significant number of services now available; message level security implemented; sophisticated management tools implemented	External and internal users managed on same basis; extensive collaborations with external businesses; significant process reengineering based on new capabilities
Phase 4 - Maturity		

Table 20 – Policy Framework

Managing Participation

From this policy framework it's pretty clear that a major factor in planning Web Services projects is the type of participation. As we show in Figure 12, there's a clear progression of complexity that needs to be managed. As the number of participants grows then one would expect that gaining consensus on the semantics of the Web Service becomes more challenging. Having said that, once standards are established then the need to gain consensus with subsequent users is of course removed.

With reference to the earlier section, Looking for Boundaries, multiple instances of any boundary, as in multiple participants, would increase the value of using Web Services as a solution.

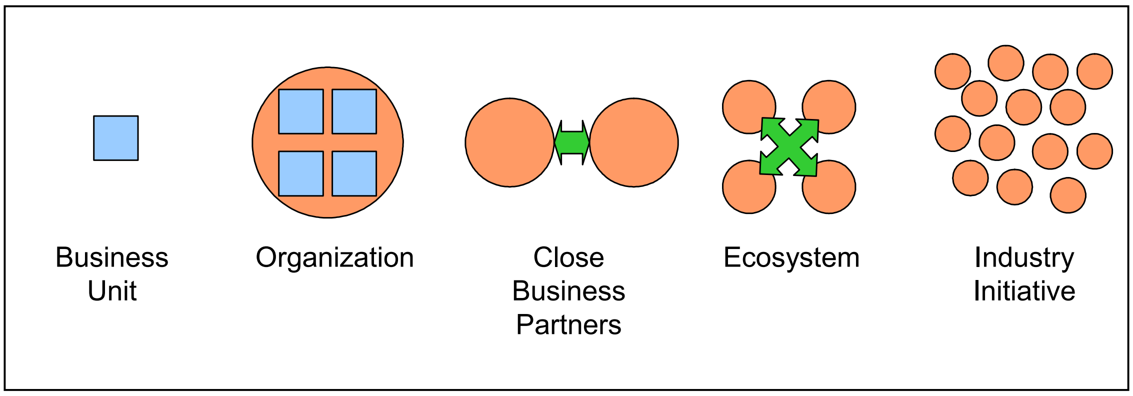


Figure 12 - Managing Participation

Whilst managing dependency itself is nothing new, Web Services make collaborations much easier and cheaper to implement, so the potential for wider participation is much greater.

Participation Horizon	Management issues	Timescale
Business Unit		Short
Organization	Normal sharing issues	Medium
Close Business Partners	Semantics	Medium
Ecosystem	Semantics and business model	Long
Industry initiative	Semantics, business model, possibly intermediary	Long

Table 21 – Participation Horizons

Roadmap Actions	
Plan & Manage	Set policy on types of Web Service application that is relevant to each phase of the maturity model
Infrastructure	Implement infrastructure commensurate with Web Service style in use
Architecture	Prioritize infrastructure activity according to application priorities, not availability of new technology
Process	
Projects	Establish application type priority and weighting mechanisms, and use in project proposal, budgeting and approval processes

Real World Migration of Development Projects to SOA

Abstract: When migrating the corporate software portfolio to Service Oriented Architecture (SOA) there are many challenges. The obvious one comes from the legacy of hardware platforms and older software models. This report explores how the special characteristics of Object Management Group's (OMG's) Model Driven Architecture (MDA) and Compuware's OptimalJ tools addresses this challenge by ensuring that new code and integration work is consistent with SOA.

Introduction

By driving all development from a model, accelerated delivery, reduction of complexity and governance follow. Governance ensures consistency between model and implementation and is a key characteristic of MDA. From it flow many improvements in the quality and cost of development, for the entire project scope including legacy and other non-service oriented functionality.

In this report we will look closely at how MDA contributes to the task of building and integrating SOA in the heterogeneous enterprise. One of the classic difficulties faced by IT Directors is that ROI (Return On Investment) cases are hard to make for projects that clean up and unify architecture. Companies like to see obvious functional improvements before committing resources to software projects. A tool like OptimalJ has a distinct advantage in its ability to accelerate business-driven projects while ensuring that the code follows best practices and conforms to SOA.

OptimalJ starts with a business model where the important objects and their relationships to each other are modelled. These objects relate to the business domain and describe items such as: people, account, product, payment, etc. Services are included in the model, typically business operations such as 'Open_Account', 'Buy_Product' and so on. Once the model has been refined, OptimalJ generates an application model and from that working Java code. Developers add Java business logic to complete the solution and the tool builds and deploys the finished components to the server.

Using MDA to Move to SOA

Managing Complexity with Model-Driven Pattern-Based Development

In our introduction we said that MDA manages complexity. This is achieved by promoting application development to a higher level of abstraction, which focuses on business requirements rather than technological details. Developers can focus their creativity on the business functionality and leave OptimalJ to build architecture.

OptimalJ builds applications for the J2EE platform. Flexibility and power have shaped J2EE into a platform that requires many lines of source code and deployment descriptors. While Compuware's wholehearted adoption of OMG's MDA specification cannot remove complexity from J2EE it can provide a mechanism for managing it through application of established patterns. Patterns are a way of capturing a tried and tested solution to a problem in a generalized format that allows others to apply it to a particular problem. Patterns have become a standard approach in the Java world and OptimalJ's MDA implementation is a logical progression from manual identification and application of patterns to a more automated method.

> *"MDA bridges what has been a significant gap between business modeling and software development by ensuring that business models drive application development, not the other way around. Patterns, however, are key. While models help designers to reduce the complexity of the business process, patterns reduce the technology complexity. The combination of models and patterns, therefore, closes the gap between business and IT"*
>
> Franco Flore, Senior Product Manager, Compuware.

The benefits of using tried and tested patterns are apparent in the quality of the resulting applications and their timely delivery.

Governance

The word 'governance' has the ring of politics about it and to understand why we throw this controversial term into the debate we must explore one common scenario facing the UML-driven team. A UML model is driven by the business problem as perceived by the analysts and business users. Once the application starts to take shape the business problem shifts in the light of experience of the prototype and a new process is needed to re-synchronize the model with the prototype code. In a large project with many developers, the model and the application drift apart and a point arrives where a reverse engineering exercise is tried. This often results in a model that is unintelligible, badly commented and laid out. So we now have a pretty model that doesn't reflect the code and a messy one that is true but unusable. Typically the analysts continue to work with their tidy model and the developers stick with the 'code view'.

Governance is our word for the ability of the tool to unite model and code so that it can't drift apart. This can be achieved best by unifying the tools used by the business modelers and the code developers. Using the extensible and open frameworks offered by IDEs such as NetBeans, Eclipse and JBuilder there is no reason why this shouldn't be achieved to the satisfaction of all concerned. Comparing the serious players in this field, only Compuware with its OptimalJ focuses on managing architecture implementation. That is to say, OptimalJ prevents the model from being changed through the Java code. This is achieved by a system of guarded and free blocks in the code. Code that implements a business requirement is added into the free blocks and this is preserved when the application is re-generated. Guarded blocks are blocked out by the code editor so that the model is never compromised by a change of the Java code.

Benefits in terms of quality, re-use and lower maintenance all follow from having a consistent model and application. Investment in the business model is protected for the future because new applications can be factored from parts or the entire model without any risk of losing bug fixes or enhancements made to the deployed application.

Changing Roles

We were interested in how Compuware see the roles of architect, analyst and developer changing where its MDA tool is in use. Clearly the system architect and analyst roles must be made easier by OptimalJ's approach but developers could be seen as being de-skilled. We asked:

"Does MDA change the developer skills needed to build enterprise class applications? Is there reluctance from developers to adopt the MDA approach?"

When using OptimalJ developers have less coding to do, as OptimalJ generates between 60% and 75% of the application. It is possible, however, to influence what type of code is generated because OptimalJ includes pattern-editing functionality enabling the customization and extension of OptimalJ's default patterns, which in turn, will influence the code that is being generated. The generated code is derived from the models by OptimalJ's Transformation Patterns resulting in pre-tested and thus high-quality code. Therefore, it isn't possible to change the generated code (protected in guarded blocks). Developers, however, have the opportunity to further define the application in the 'free blocks'. MDA compliant tools should provide a mechanism where developers can add their code to the generated code. This allows them to focus on the real added value of the application, which is adding business logic and they don't have to bother about specific low-level coding details. This allows developers to continue being creative and to focus on adding value by enhancing business logic, rather than building infrastructure. In this scenario developers should not be reluctant to adopt MDA but should see the benefits of embracing it.

Building the Business Case

Enterprise IT is emerging from a period in which central controls were relaxed in order to allow the business benefits and not system architecture to drive purchasing decisions. With these constraints it was difficult to impose any architecture on the collection of point solutions and packages that inevitably built up. Is it likely that a realistic return on investment case could be made for cleaning up the architecture?

We recommend that the only way to combine development productivity with a timely transition to SOA is to use tools that give you the architecture at no additional cost; i.e. with no extra development resource. MDA is a good candidate here because it can deliver functionality quickly, guarantee SOA in its new solutions but most importantly OptimalJ can integrate exiting resources into the new architecture, again with minimal cost.

OptimalJ: Management Overview

OptimalJ is a full implementation of OMG's MDA specification; unlike the previous generation of code generators it is standards based and generates J2EE code.

Domain Model, Application Model and Code Model

OMG's approach describes separate models; the Platform Independent Model (PIM) and the Platform Specific Model (PSM), which maps to code to create a viable application. OptimalJ implements these two models as the Domain Model and the Application Model, and maps the code to a Code Model. OptimalJ uses a set of rules known as Transformation Patterns to carry out transformation from one model to the next.

MDA's models force a separation of concerns as follows:

Platform Independent Models (PIMs) provide formal specification of the structure and function of the system and is independent of the computing platform. In OptimalJ this model contains the business objects and services. OptimalJ provides a subset of the full UML capability allowing analysts to specify business objects and services. Models can be imported from other UML tools.

Platform Specific Models (PSMs) are generated automatically from the PIM using OptimalJ and contain components that apply to the target platform and architecture. In this model the technical architecture is now visible at the level of components, web pages and services.

Code is generated from the PSM and includes all the source code, Java, deployment descriptors, web pages, SQL scripts and so on to run the application.

MDA describes the mappings needed to transform one model to the next and also describes the refinement and reuse of components in both the PIM and PSM. MDA builds on existing XML standards, for example UML uses the XMI (XML Metadata Interchange) standard to communicate structure of objects and interfaces and MOF (Meta Object Facility) is used to describe objects.

OptimalJ sits very solidly in the analysis, design, build and re-factor stages of the application development lifecycle. It leaves the infrastructure provided by the J2EE application server to sort issues of management, security routing etc. This should be seen as a clear benefit, especially in a time when the full standards stack is still emerging and no one wants to build any dependencies on non-standard server capabilities. Most would see that the lack of any 'black boxes' in the runtime implementation of an OptimalJ application as a 'must have' feature. After all why pick an open platform such as J2EE if you then become dependent on a 'closed' module from a tools vendor. Once you have a well modeled and 'blue print' Java application the down-stream benefits flow from the flexibility, re-use and modularity.

Using OptimalJ to Move to SOA

To briefly re-cap, OptimalJ's key strengths are:

- Business focused development where key development resources are building business functionality not architecture
- Automatic application generation from a high-level business model leading to accelerated development
- The model includes the concept of services that promotes the design of a consistent and business-focused service bus.

- Reduction of complexity and improved quality through application of industry-standard patterns
- Governance of process ensuring a consistency of model and code leading to improved re-use, lower maintenance and improved quality.

Modeling Services

A service model consists of a set of business operations with consistent naming, input and outputs and relationships to the business objects they affect or use. We believe that the full benefit of SOA is possible only when you have a consistent model and a business overview of services. The Domain Model in OptimalJ includes the Domain Service Model where you model business applications. Once you have the model, existing software assets can be integrated into the SOA and new functionality built where is doesn't exist.

Enterprise SOA

Now let's look at the real world of enterprise applications and in particular whether the MDA paradigm and MDA-based tools offer any advantage when faced with a mixture of packaged applications that developers can't touch, old client-server applications, a few desktop applications based on Excel and Access, a small Java web application, and some mainframe applications with traditional CICS APIs or similar. Not much sign of the Business Services Bus here! So you have the latest copy of OptimalJ - where do you start?

We can identify a number of distinct steps in the transition. Step one is 'Model', which includes 'Inventory' and 'Map' and steps two, three and four are 'Build', 'Integrate', 'Migrate'. In the first step we are designing architecture for the business and creating an inventory and in steps two, three and four we create new services, integrate the existing applications and begin the migration towards SOA. By generating code that implements SOA the developers concentrate on building functionality that will be visible to the business users.

The following table sets out some key activities we identify on the roadmap to SOA. Some activities would be completed using OptimalJ's model, but others such as the inventory and mapping activities could be done with office tools.

Steps	Description	Tool/Formats
Model	Build a UML business model, including a service model	OptimalJ or other modeling tools
Inventory	Document each application in the enterprise, breaking each one down into domains that map loosely to the business model	HTML document with links to UML use cases and business model, use case model
Map	Identify services provided by each application and map each one to the service model, including a traffic light status reflecting whether the service can be re-used in the proposed services bus	HTML document showing the relationship between the existing and proposed services
Build	Generate the service bus code from the model as session beans with Web Service interfaces	OptimalJ, MDA tools
Integrate	Use the generated classes as wrappers for existing functionality identified from the inventory	Connectors based on JCA (Java Connector Architecture), CORBA, Web Services
Migrate	Replace inflexible code with components that fit with the planned SOA	Development platform IDE

Table 22 - Key Activities on the Roadmap to SOA

Achieving SOA inevitably involves some enterprise integration work, which calls for an adapter approach. Fortunately J2EE has embraced this requirement with JCA. So the integration work can be modeled and documented in the same tool set.

JCA Integration Model

OptimalJ implements the JCA architecture and utilizes the JCA resource adapters provided for platforms such as Cobol/CICS and IMS/DC. JCA supports the concept of contracts and can handle issues such as security and transactions in the J2EE application server. Its modeling capability extends to the Connector model that can be manually entered or imported from Cobol copybooks.

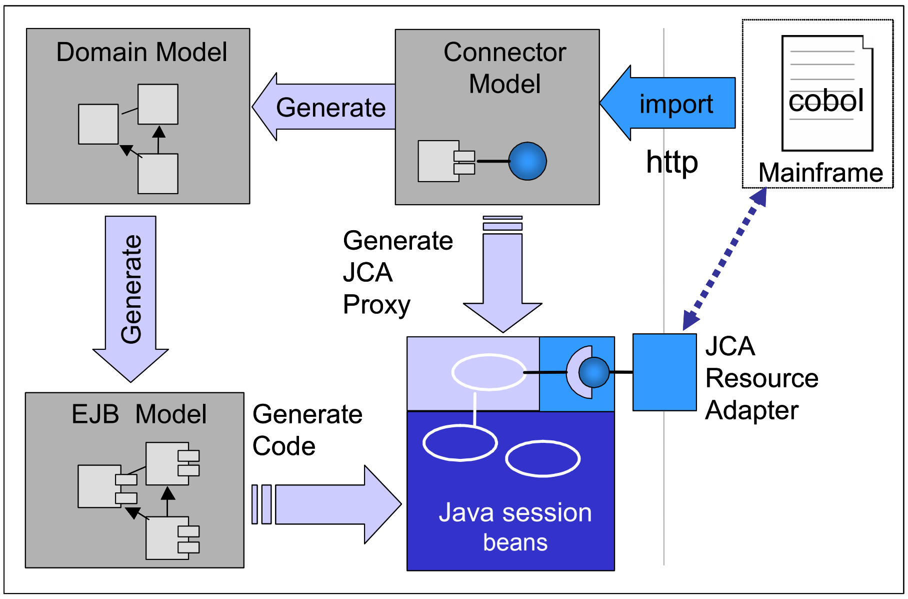


Figure 13 - Integration Model

The same approach works for CORBA IDL or of course Web Services, which is shown below. UNIFACE, Compuware's 4GL development environment has an established user base who can also benefit from integration capabilities that OptimalJ brings. UNIFACE uses the CORBA interface to expose functionality and will be of particular interest to these users. Additionally with the latest UNIFACE release, UNIFACE Services can be transformed into Web Services. In both cases UNIFACE automatically generates the interface definition (IDL or WSDL) which can be imported into OptimalJ to create a Connector Model, which can be used to populate the Domain Model; we believe that this ability to build a service and integration model is an important capability that ensures that a coherent migration to SOA is possible.

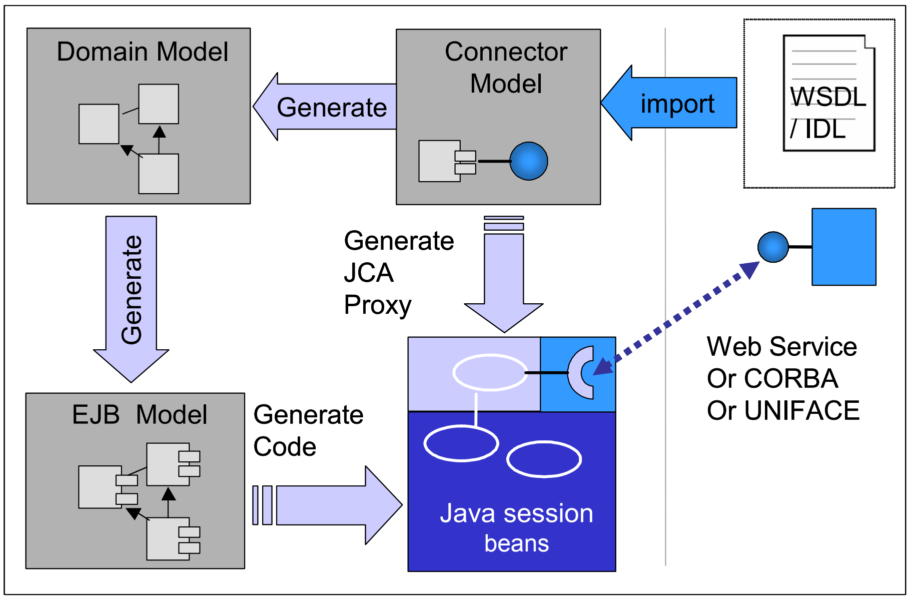


Figure 14 - Integration Model for CORBA and Web Services

Web Services Support

Web Services provides the platform independence required by the heterogeneous enterprise. The service layer generated by OptimalJ can be selectively published as Web Services simply by selecting a component in the model and choosing a menu option. There is no need to change any code or write any configuration files because OptimalJ's code generator step deals with all the deployment issues for your selected platform.

Model-based Pattern-Driven Code Generation

Finally we look at OptimalJ's code generation capability and focus in on what makes this code generator so different from less successful forerunners.

The key features of Compuware's new approach are:

- UML platform independent business model (drives the whole process)
- Transformation patterns and customizable implementation patterns (model-to-code transformation)
- Active synchronization
- Business rules
- Integrated deployment
- Standards-based methodology and code implementation

OptimalJ promotes a top-down approach by driving the code generation from the UML business model and maintains the model when the resulting code is customized to implement business logic. Unlike earlier generators however, the code that is generated follows an established set of patterns and best practice as documented in Sun's J2EE blueprints.

Active synchronization removes the need to reverse-engineer the model from the Java code; the IDE used by the developer is integrated with the model and 'knows' which lines of Java the developer cannot change (so called guarded blocks). Any changes to the model must be made in the model. Free blocks are designated for the developer to plug in the business rules and processing logic.

The patterns used to transform the Domain Model into an Application Model and then to code are not closed to the system architect and can be altered in order to follow company standards. OptimalJ's Template Pattern Language (TPL) is used to define the transformation from model to code. Join points add more subtlety to the pattern customization by enabling a join to be made between a customized pattern and the pre-configured pattern supplied by Compuware – thus you can update your OptimalJ patterns without losing your special enhancements.

Summary

In this report we have looked at the issues facing a company moving towards SOA from a heterogeneous set of applications and platforms. We recommend a process that starts with a top-level service model into which new and existing functionality can be integrated. By focusing on how OptimalJ matches up to these types of demands we have identified some of the key features of OptimalJ:

- Service model
- Automatic generation of an application model with SOA
- Support for enterprise integration within the model
- Support for Web Services

MDA is a powerful way to manage the complexity of distributed computing platforms such as J2EE and OptimalJ's approach takes pattern use to a new level of productivity, addressing SOA and Web Services through its consistent application of patterns. Early adopters of OptimalJ report productivity gains using OptimalJ but we note that the service architecture is taken as a given. The issue isn't anymore what is SOA or do we need SOA, but how fast can we get there?

Links

OMG MDA	Model Driven Architecture MDA Documents
	ormsc/01-07-01: Model Driven Architecture (MDA)
	http://www.omg.org/cgi-bin/apps/do_doc?ormsc/01-07-01.pdf
	omg/01-12-01: Developing in OMG's Model Driven Architecture (MDA)
	http://www.omg.org/cgi-bin/apps/do_doc?omg/01-12-01.pdf
Compuware	http:///www.compuware.com/
Success Stories	http://www.compuware.com/products/optimalj/1792_ENG_HTML.htm

Web Services Roadmap for the On Demand Business

IBM's strategy today is centered around "Business On Demand" in which their entire set of products and services are focused on delivering greater business efficiency and agility. A core element of the strategy is the transition to Web Services, and in this section we examine how IBM is turning this high level goal into reality for its customers through practical delivery of products, services and Roadmap guidance that enable the on demand business.

On Demand Business

IBM has initiated a strategic business change which over the next five years will have profound impacts on how it engages with its customers. This change is a move to provide end to end business process support to their customers, which includes the entire life cycle of a business process, spanning business design through to operational management.

By on demand, IBM means the delivery and execution of a business process when and where it is required in the most efficient and cost effective manner. On demand encapsulates a number of fashionable business trends. First, the on demand business needs to be a **Real Time Enterprise (RTE)** where events are responded to as they happen. This requires more real time systems behaviour. No more weekly updates or working off yesterday's out of date information. Acting in real time often requires **Straight Through Processing (STP)** to remove time wasting steps (typically human) from the process. We can argue about just how real time things need to get, but there is little doubt that optimizing the enterprise's use of resources through **Just In Time (JIT)** approaches are beneficial to the bottom line. But JIT requires **Business Process Optimization (BPO).** There is little point in implementing JIT Manufacturing for example if the sales order business process and inventory management are not brought into line. However, few businesses are an island, and they work with many partners up and down the supply chain. So BPO more often requires a broader look at **Supply Chain Optimization (SCO).** Each of these business buzzwords is highly related and dependent. It is hard to deliver one without addressing the others.

On demand also means dealing with peaks and troughs, constantly shifting requirements, and requires the agility to not just transition to this new world on a one time basis, but to constantly optimise processes and use of resources. To meet this challenge, organizations will increase levels of outsourcing, yet must at the same time demand higher levels of integration from their partners to deliver STP and SCO. Businesses, and their processes, will become increasingly virtual so that they can constantly reconfigure themselves on demand.

Primarily, this will require greater levels of automation of business processes. If they are to be executed quickly, they require minimal human intervention. Achieving the on demand business is therefore dependent on information technology. But for the IS department to meet the businesses needs in the future it too must undergo a similar transformation. Cycle times to deliver new systems or updates must be drastically reduced. Capacity must be available JIT. Down times or interruptions in service must be averted. In other words, IS needs to become an on demand business too, and requires just as much BPO as any other part of the business.

As such, we can view on demand from two complimentary perspectives which we can term,

On Demand Business Processes. The business processes behaviour outlined above.

On Demand Operating Environment. Providing an IT and IS infrastructure that delivers resources on demand, efficiently, and cost effectively.

The IBM On Demand Operating Environment

IBM defines the on demand Operating Environment as a “flexible, open, integrated infrastructure for rapid deployment and integration of business applications and processes, virtualization of resources, and for automated, resilient systems”. As most of IBM customers’ systems are built in a heterogeneous platform, it is based on open standards like Web Services and the OGSA framework. There are three key capabilities:

- **Integration** creates business flexibility and the collaboration between people, processes and information by combining disparate, unconnected data, applications and processes.
- **Virtualization** improves the utilization of IT, information and people assets because it allows customers to treat resources as a single pool, accessing and managing those resources across their organization more efficiently, by effect and need rather than physical location.
- **Automation** improves availability and resiliency, while reducing the complexities of managing IT and lowering IT management costs, based on the policies set for the business.

The IS Department Challenge

Desirable as the on demand business may be, many enterprise customers will however still have what they perceive as more mundane requirements, and will see the challenge of delivering an on demand business as considerable. For example

- Many will still be focused for example on internal Enterprise Application Integration (EAI) projects. EAI alone doesn’t deliver the on demand business. Can they solve EAI and on demand requirements in one go?
- Besides EAI, large enterprises will have been presented with a number of options in recent years that additionally address B2B and distributed computing requirements, each delivering incompatible solutions to the underlining problems of integration. Is on demand going to require another different technology solution?
- The reuse of existing assets is key. Though they may not all deliver optimal support for the on demand business, the cost and time to replace them is too high. Can they be reused yet again?
- The typical IBM customer will also have a heterogeneous environment that mixes not only diverse IBM platforms but those from several other vendors too. Will on demand require yet another layer of software infrastructure to be delivered onto every platform?
- Are the skills required difficult to acquire? Is the learning curve to get developers up to best practices steep?

Web Services are Key

IBM's vision for on demand business is critically dependent upon Web Services. Though aspects of on demand could be delivered through a diversity of niche (and typically proprietary) technologies, it is clear that Web Services will play a central role. For example

- It provides a single ubiquitous messaging/communications infrastructure based on open standards, that should be common to all participants in on demand business scenarios,
- Web Services provide high levels of automation to the solution delivery process. Automation is going to be central to on demand business. Not just enabling automation of the business processes in support of the business, but automating the development, discovery and usage of Web Services by the IS department.
- Provides mechanisms for publishing Service descriptions and dynamic Service discovery, which facilitates on demand assembly of solutions and aggregation of new Services

- Is being used at the core of autonomic and grid computing approaches which will provide amongst other things resilience and scalability in on demand business implementations
- More specifically, there are emerging Web Service protocols that enable the operation of the on demand business
    - BPEL – dynamic location of business process execution
    - WS-Addressing – redirect service requests. For example to dynamically relocate implementations
    - WS-Security – provides federated security support
    - WS-ReliableMessaging – guaranteed message delivery
    - WS Distributed Management – support SLA in an federated on demand environment

Importantly, Web Services provides solutions for more immediate requirements of enterprise customers, and with careful application can provide an infrastructure that enables a transition path to full enterprise-wide on demand business implementation. Many customers will see some of their current projects as steps towards on demand business with a focus on enabling some aspect of the various business buzzwords outlined earlier. However, basing those projects on proprietary technologies without the application of Web Services, and more importantly without taking a SOA approach, is likely to mean those solutions will have to be reengineered in future.

Consequently, Web Services are increasingly core to IBM's strategy, helping them deliver solutions to customer current and future needs. Web Services hide the diversity of the platforms and systems behind a standard interface, enabling new solutions to be assembled on demand from existing and new assets regardless of their implementation.

IBM's leadership role in driving towards open standards for Web Services is well known. As key contributors to, and usually instigators of the majority of Web Service protocols together with Microsoft, IBM is in a strong position to influence the standards process to meet the requirements of their customers and deliver early support in their products and services. We believe that open standards will be essential to free the on demand business from the constraints of otherwise being locked in to vendors proprietary solutions.

Besides their central role in core Web Service protocol standards, IBM have also formed a set of Web Services Industry Councils (WSIC). These address the specific needs of different vertical industries, where for example increasing B2B integration will drive business partners towards greater consensus on the semantics of the information they exchange and the collaborative processes they share. The initial focus will be on financial services, manufacturing, distribution and retail, and public sector.

Components of Web Services Success

For IBM's customers, successful implementation of Web Services will likely revolve around three key areas

- Web Service enabled products. Few new classes of products will be required to support Web Service delivery. However, upgrades to the latest releases or extensions are likely.
- Application of Best Practices. Not only providing a quick start though patterns and frameworks, but also encapsulating the experience of IBM architects and consultants
- Support of Professional Services. Providing additional resources, ready skilled in Web Services.

Software Products

As stated earlier, Web Services are core to IBM's software product strategy. The portfolio of products spans five brands which IBM is constantly upgrading to support the latest Web Service protocols and concepts. An overview of the current support for Web Services offered by IBM products is shown in Table 23.

Brands and Key Products	Web Service Capabilities
DB2	
DB2 Universal Database V8	XML Extender generates Web Services to query and update table data Publish stored procedures as Web Services
DB2 Information Integrator 8.1	Compose, transform and validate XML documents and data
Lotus - Provide collaborative Web Services	
Lotus Domino 6	Subsumes WebSphere Application Server Web Services support Enable Web Service based collaboration in majority of IBM Lotus products including Notes, Domino, Workflow and Discovery Server.
Rational	
Rational Rapid Developer	Architected RAD based Web Service creation
Tivoli - Managing deployment and operation of Web Services	
Tivoli Configuration Manager	Installation and configuration of Web Services
Tivoli Access Manager	Centralized policy management of Web Service applications
WebSphere - Develop, host, deploy and publish Web Services	
WebSphere Application Server V5	Private UDDI Registry for publication Web Service Gateway helps make internal Web Services available to a wider variety of consumers, both internal and external Web Service Management (WSM) capabilities Web Service Invocation Framework (WSIF) supports variety of transports WS-Security support Workflow with Web Services
WebSphere Studio Application Developer Integration Edition V5	Create Web Services from software assets such as JavaBeans and EJB, JCA adaptors, etc Service Flow Editor Workflow/Orchestration support
WebSphere Business Integration	Web Service Connectors for WS Gateway, WSIF, process and message based WS connections Web Services Application Adaptors
WebSphere Portal	Support Remote Portlet Web Services (predecessor to OASIS WSRP)
WebSphere MQ 5.3	Assured delivery of Web Services using MQ transport Provide and Consumer business processes activities as Web Services
WebSphere Commerce	Subsumes WebSphere Application Server Web Services support
WebSphere SDK for Web Services 5	Self contained tools and deployment environment for Web Service environments Support for latest Web Service protocols

Table 23 - Web Services Support from IBM Products

Alphaworks

Also of particular interest are the technology previews available via the IBM Alphaworks site that provide an early opportunity to examine a number of emerging tools, applications and frameworks that exploit Web Services protocols. Thought these cannot be used in production, we recommend looking at the technologies like those listed in Table 24. These demonstrate that IBM is thinking beyond the basic Web Service platform provision and providing valuable functionality that enables the assembly of on demand infrastructure and applications.

Technology	Web Service Capabilities
Web Services Outsourcing Manager (WSOM)	Framework that enables dynamic on demand composition of Web Service based business processes.
Emerging Technologies Toolkit (ETTK) Formally Web Services Toolkit	Software development kit contains many utilities and tools for designing, developing, and executing Web services, as well as emerging autonomic and grid-related technologies.
Utility Web Services (in ETTK)	For example, User Profile, Metering, Accounting, Contract, and Notification.
On Demand Service Grid (in ETTK)	On Demand Service Grid: service broker that manages a heterogeneous group of service suppliers to provide services to multiple groups of consumers
Web Services Bus (in ETTK)	Supports both the service requestor and service provider roles in a service oriented architecture. The Bus promotes separation of business logic from infrastructure, and provides format and protocol independent deployment and invocation of web services. This enables services exposed by a number of different component types to be used in a uniform manner.
Web Services Tool Kit for Mobile Devices	Provides tools and run-time environments that allow development of applications that use Web Services on small mobile devices

Table 24 - Some Alphaworks Web Service Technologies for on demand

Delivering Best Practices

jStart

jStart is a team of IBM experts who help IBM's customers implement emerging technologies. The jStart program originated to support Java and now focused on Web Services. jStart commences activity early in the lifecycle of any new technology and often takes technology direct from the IBM labs and gives essential feedback to the labs on real world implementation. Consequently, both customers and IBM want to reduce risk and make sure their Web Service projects are successful. Partnership is seen as essential, and there must be mutual benefit to all.

The program follows an engagement model illustrated in Figure 15. The jStart team work with business users and senior IT staff to stimulate thinking of how to use technology to benefit business and create opportunities where Web Services might best be applied. The jStart team seek to address real business problems – pure technology pilots are not of interest. The goal is to deliver success that is recognised by the business. In return, the business must agree to the publication of a reference case study.

At the end of phase III the decision to turn the project into reality must be made. At this point, a Service Oriented Architecture (SOA) will have been developed and 3 or 4 use cases that will be the basis of the Web Services are investigated in detail to scope out a pilot project. Working from the beginning to end, a typical project takes 45-60 days.

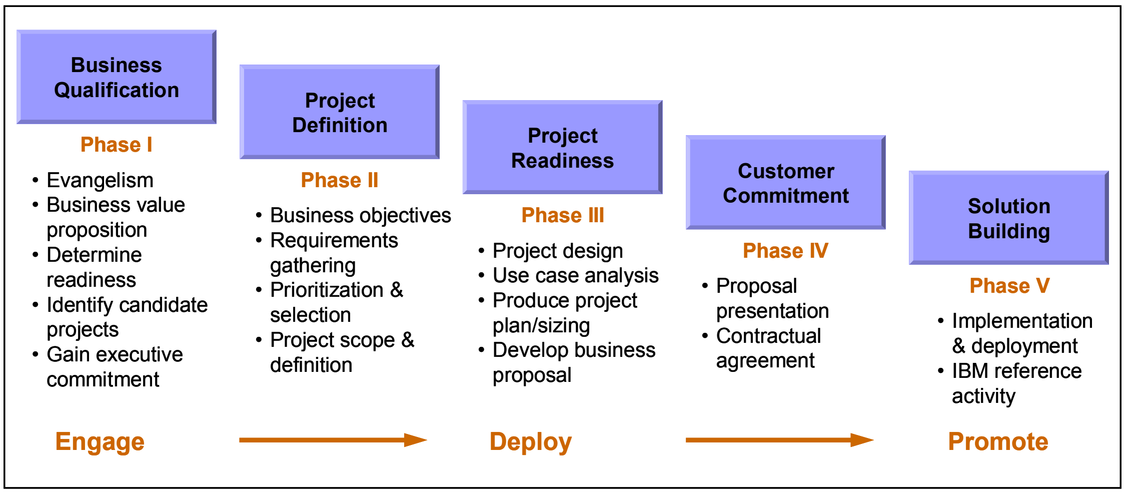


Figure 15 - jStart Engagement Model (courtesy of IBM)

Today, jStart find that the typical project is focused on improving efficiency (as opposed to introducing new business ideas). In the current climate, projects to automate existing manual processes are popular for example.

IBM continues to improve the jStart program. For example they often find that on completion of the pilot due to its very nature the technology as already moved on. To address this they have now added a two day Web Service review workshop with the customer to consider the latest developments. At the time of writing that might mean considering WS-Security, BPEL, or the latest IBM tools. As well as increasing the focus on security during the rest of the year now that standards are stabilising, jStart are also working with customers to address Web Service provisioning. I.e., the delivery of commercial Web Services, Service Level Agreements, Web Service Management approaches, and other issues related to delivering Web Services on a commercial basis.

The jStart program certainly appears to be working, with an impressive number of Web Service case studies already published on the jStart site.

IBM Patterns for e-business

IBM has developed a number of patterns reflecting common e-business scenarios that provide an excellent starting point for IT architects. Many of the patterns have been documented in detail in the associated book, Patterns for e-business: A Strategy for Reuse, and in various Red Books available on-line.

IBM is now in the process of updating these patterns to reflect the use of Web Services where applicable. At this point no new patterns are envisaged but several existing ones, such as Extended Enterprise (B2B) or Application Integration, are being updated show where Web Services might be applied, and how they compare to alternative technologies that could be used. For example, they will provide guidance on delivering Service Level Agreements, Quality of Service, and security.

This is a useful first step in support of what we term “conversion” scenarios, i.e. applying Web Services to current application architectures. In future we would expect to see further patterns emerging to support “exploitation” scenarios, i.e. new on demand architectures that are only really feasible once the Web Service infrastructure is in place. For example, the dynamic discovery of new service providers, or support for autonomic computing.

Speed Start for Web Services

At the developer level, IBM has introduced the Speed Start for Web Services training and guidance program. This makes available to them trial versions of the latest IBM software development tools and middleware, online tutorials and articles, hands-on workshops and technical briefings, and an online forum moderated by IBM Web services experts.

Web Services Roadmap

Enterprises will not transition to on demand business overnight. IBM's enterprise customers need an evolutionary architecture that will get them there in stages, and address some more immediate problems along the way.

We see the roadmap consisting of two parallel but entwined tracks. Ideally, the provision of an on demand operating environment should to a large extent precede the transformation to on demand business Processes. Having said that, nothing stops new Web Services and business processes being designed with on demand business in mind but implemented in today's operating environment. Circumstances will often dictate this. The key is to design and implement them in such a way that moving them to an on demand operating environment at a later stage is as painless as possible. As such, the use of Web Services and adoption of SOA principles will be essential. In the following section we asses these stages according to the CBDI Web Services Maturity Model.

Early Learning Phase

Organizations will usually commence with tactical, ad-hoc use of Web Services to meet immediate internal and external requirements. These would not typically be implemented today as part of a concerted effort to put a foundation in place for the on demand business.

Nevertheless, some useful steps can be taken. For example, exposing external services can make an organization look more responsive particularly where it automates what was previously a manual task for the service consumer.

At this stage partnering with IBM in the jStart program might be the most appropriate way to not only address some immediate Web Service needs, but to ensure best practices are adopted and that the correct first steps towards on demand are made. We would also recommend starting to get developers involved as soon as possible in the IBM Speed Start for Web Services training and guidance program.

Integration Phase

In terms of the immediate future, the integration state is probably the most important. At this stage, most activity will revolve around optimising existing business processes where improving integration and accessibility will be the key drivers. However, to meet these and other goals of the on demand business it is essential that Business Services have been developed with the longer term in mind. Business Services need to be properly abstracted away from current implementations.

During the Integration phase, IS should be laying the groundwork for on demand. Steps include

1. Start implementing an enterprise wide Web Service infrastructure. Upgrade application servers, middleware to support Web Services. Install Web Service Management capabilities to manage SLA. Provide an internal UDDI registry.
2. Expose Web Services from existing applications. Owners of applications should convert their existing interfaces to Web Services. From a technology perspective, this is often a straightforward usage of the platform or packaged application vendors latest Web Service aware releases of their products on which the existing apps are based. Web Services can then be used as an open-standards approach to EAI in the continuing effort to optimise internal processes.
3. Delivering Web Services based on SOA Principles. Apply Service Oriented Architecture (SOA) principles in development and service design. For example, carefully abstracting the Service away from current implementation(s).
4. Implement a Business Service Bus approach. The Business Service Bus exposes Web Services that reflect meaningful business concepts to service consumers. The 'bus'

groups together related Web Services that will share common elements of specification and taxonomy in a specific business domain

5. Adoption of SOA and use of the Business Service Bus will be a core here as these that will deliver the long term flexibility that the on demand business needs. Simply exposing Web Services directly off existing systems can be suboptimal as they can reflect too closely the existing implementation, and are more often affected by changes to that. Additionally, existing APIs are often of the wrong granularity, particularly for external use, and Web Services exposed in this way can leave too much work for the service consumer (both internal and external) to aggregate and refine them into something useful.

This does not imply that Web Services should not be exposed from existing services, rather that steps 3 and 4 above be used to create a two layers architecture that separates what we term implementation based Web Services from the more meaningful Business Services that are used by the Service consumer. Benefits of this approach include

- Truly hides the implementation from the service consumer
- Enables on demand aggregation and composition of new Business Services from implementation based services
- Provides transition path to the on demand Operating Environment, enabling Service implementations to be outsourced and dynamically switched with a minimum of impact on the consumer

IBM provides a number of technologies, primarily as part of the alphaworks program listed above that facilitate this approach, including the WS Invocation Framework and WS Gateway, both of which are contained in their Web Services Bus. Together these (with other IBM middleware) enable the delivery of implementation based Web Services and the mechanism to aggregate them into, and manage them as, Business Services.

IS should also be re-engineering itself at this in preparation for the re-engineering of the business. There is little point in trying to deliver an on demand business, whilst IS still has long lead times. This does not just mean putting the on demand Operating Environment in place. IS needs to consider applying on demand principles to,

- The way in which systems and their components are analysed, designed, assembled and built
- The availability of human resources, and appropriate skills
- Willingness to use external Service Providers, Hosts and various intermediaries, both as sources of Web Services, and to enable the delivery of Web Services

Re-engineering Phase

The Integration Phase should see the Web Service infrastructure in place, well formed Business Services, and existing business processes optimised. At that stage, BPO will primarily be the replacement of existing interfaces (human or machine) with Web Services to deliver STP. However, often the business process itself will not have changed much

Though it might not be a popular term, some Business Process Reengineering is now inevitable. Similarly, it is at this stage that the IS department should be “eating its own dog food”. That is, if on demand is good enough for the business, it should be good enough for IS and it should start widely using the on demand operating environment. For example

- At this stage the location of data and computing resource should not matter, providing adequate security and SLA is in place. This does not just mean that data is stored off site, but that data is retrieved from the owner on demand as needs require rather than replicated into the organizations own database.
- Business Processes will be operated in a more federated and parallel approach, rather than the sequential supply chain approach of today. Events will trigger multiple business activities across the business process ecosystem.

- Fine grained Business Process outsourcing will be possible – i.e. outsourcing the execution of certain individual steps, not the whole process, to get availability of appropriate resources on demand. Web Services will ensure these processes appear seamless regardless of the location of execution of each step.

We expect that key emerging Web Service technologies that IBM is promoting will be adopted at this stage to enable these activities. For example

- Business Process Execution Language (BPEL) – Provides a more dynamic approach to implementing business processes. Process steps can execute wherever there is an appropriate engine.
- Web Services Distributed Management (WSDM) –mechanism for managing SLA across distributed Services

Maturity Phase

One hesitates to write and comment about maturity because it is high probability that by the time we ever reach the mature state, new concepts will have superseded what we are working with today. However at this stage services are ubiquitous and the on demand business is a reality. Federated services collaborate and create complex products with individual services provided from potentially many providers. The capability offered by Web Services to the mature on demand business is illustrated in Table 25.

On Demand Business	Web Services Status
Real Time Enterprise	All core business processes are offered as Web Services with real time execution and currency of data;
Straight Through Processing	All core business services have been reengineered to minimize intervention, but also to establish comprehensive business monitoring and measurement controls and highlight exceptional behavior
Just In Time	A complete inventory exists of core business services that allows existing processes to be altered and new products, processes and channels to be introduced with minimum time and cost.
Business Process Optimization	Web Services automate process flows eliminating wasteful self-service and other manual activity including external process steps.
Supply Chain Optimization	Accurate and timely data is retrieved on demand from owners in the supply chain via Web Services rather than replicated across it. Shared, collaborative Services enable more parallel activity in processes rather than sequential.

Table 25 - Web Services for the Mature On Demand Business

Timeline

During 2003, we expect most users to remain in the Early Learning phase. Important preparation for the Integration Phase should also commence in terms of training and infrastructure upgrades.

Though external Web Services will be commonplace during the Early Learning and Integration phases, the prime focus at this point will be on enabling on demand business processes and Operating Environment from an internal perspective. Externalisation of these will not happen widely until the Reengineering phase.

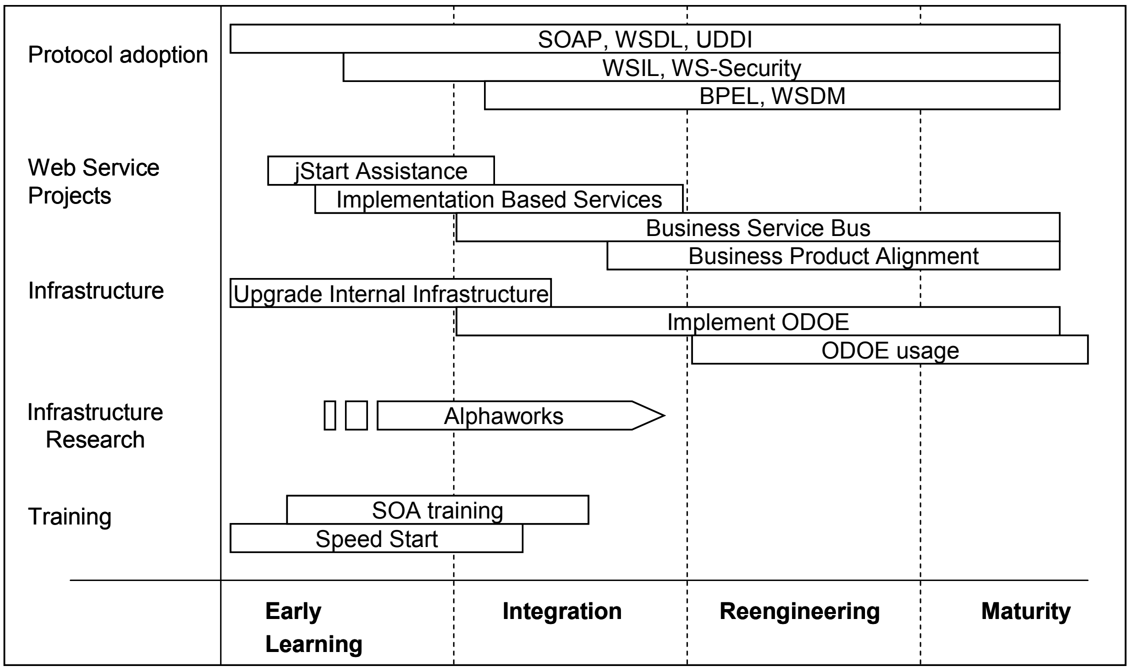


Figure 16 - Adoption Timeline

Finally, let's ask if the challenges outlined earlier will be addressed.

• Can they solve EAI and on demand business requirements in one go?	Yes. Both EAI and on demand business can be addressed by Web Services. Addressing current EAI via Web Services provides a better transition path towards on demand business
• Is on demand business going to require another different technology solution?	No. EAI, B2B and distributed computing needs can all be addressed via Web Services and provide a common infrastructure to support on demand business
• Can existing assets be reused yet again?	In the Integration Phase yes. Much of the existing infrastructure has already been Web Service enabled to support this. Leading package vendors have enabled their applications too. However, the modus operandi of existing application may not be optimal for the Reengineering Phase.
• Will on demand business require yet another layer of software infrastructure to be delivered onto every platform?	Not really. Much of the existing software infrastructure will need to be upgraded to the latest versions to support Web Services, but a new additional layer shouldn't be required.
• Are the skills required difficult to acquire? Is the learning curve to get developers up to best practices steep?	On the whole no. The technology of Web Services will be largely transparent to developers. However Service analysis and design will require some rethinking. Best practices encapsulated in templates and frameworks will help considerably.

Summary

IBM provides a comprehensive set of Web Service enabled products and service offerings that enable their customers to implement an on demand business. Importantly, IBM is making sure that existing customers can (once again) take existing core technology investments forward via Web Service support for technologies such as CICS, MQSeries, and DB2.

Though the vision isn't necessarily completely new, we are impressed by the depth and breadth of the research that IBM is putting into making what is essentially the next generation of IT a reality. On demand pulls together many threads that IBM is at the leading edge of, such as Web Services, Pervasive, Autonomic and Grid Computing. However, individually these are technology centric messages and consolidating them in an on demand business message that demonstrates greater value by them working together to solve business problems should be clearly more attractive to the business user. Besides the obvious benefits of the business buzzwords as highlighted at the beginning, businesses are quite used to placing dependencies on external agents, i.e. in terms of creating supply chains and outsourcing non-core processes, and moving to JIT. As such they should be very receptive to and understanding of an on demand message

However, whilst IS organizations appreciate these benefits, and can also see the opportunities created by an on demand operating environment within IS itself, in our experience when discussing the use of external Web Services, they currently have greater concerns regarding making run-time dependencies on external agencies. This impacts the implementation of both on demand business processes and on demand operating environment.

But the transition to on demand business is not going to happen overnight. This is something that will take a number of years, though it will in our belief happen. As such, IS should not dismiss the externalization of on demand based on current market immaturity. Instead they should be commencing now to put place the necessary infrastructure and practices to support the transition to on demand business as it evolves. Partnering with IBM through initiatives such as jStart would be one recommended course of near term action.

Links

IBM Web Services Home	http://www.ibm.com/webservices
IBM Developerworks	http://www.ibm.com/developerworks/webservices
IBM Alphaworks	http://www.alphaworks.ibm.com
IBM jStart	http://www-3.ibm.com/software/ebusiness/jstart/
IBM E-Business Patterns	http://www-106.ibm.com/developerworks/patterns/
IBM Speed Start	http://www-106.ibm.com/developerworks/offers/ws-speed-start/
IBM On Demand Operating Environment	http://www-3.ibm.com/software/info/openenvironment/

Microsoft in Transition - Delivering a Less Complex Service Oriented Platform

Abstract: One of the difficulties when writing anything about Microsoft is the number of strongly held preconceptions held about the company. Everyone has Windows and Microsoft Office and everyone believes they understand the company's strengths and weaknesses. Indeed some journalists have written that .NET is more a marketing angle than a new technology platform. Our readers know better. The Microsoft that re-engineered a completely new platform in order to do Web Services properly and to meet the needs of collaboration and SOA is a very different organisation that refused to get excited about the internet way back when. For this Web Services Roadmap report we examine three core Microsoft strategies that are driving the transition from desktop specialist to SOA platform provider. We will look at the strategy to reduce complexity for developers and the increasing recognition that system architects need support and that there may be architecture issues beyond which language to write in.

Technology Platform and Complexity Reduction

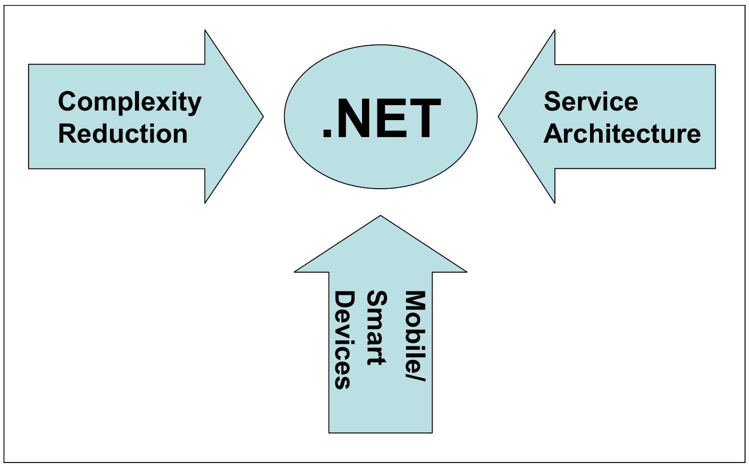


Figure 17 - Microsoft .NET

.NET launched when spending levels in IT were on a downturn and cost justification became even more rigorous. We asked Microsoft to explain some of the background to the .NET project in Microsoft.

"Today we are in a situation where complexity is consuming a large element of the IT budget, with over 70% of this being spent on sustaining and running existing systems. And only 30% or less being applied to new systems that facilitate business transformation. The current economic climate only serves to really focus people's minds on this issue. We have to turn this around and move towards a situation where this is nearer a 50:50 split, creating an environment for IT to deliver business agility. To achieve this vision we're make huge investments in underlying the architectural tenets of all of our software to reduce the overall complexity that occurs when you bring the many pieces together to build, deploy and manage a business solution. Our vision is to provide the fastest and easiest way to build, evolve and orchestrate connected applications.

Further more business process typically does not respect the artificial boundaries we place between our people, either by hierarchy or in the form of the information tools we give them (or not). Technologically it is within our grasp to represent business processes electronically through the deployment of fixed line and wireless networks, new form factors of PC hardware and a common set of open Internet standards. However it is at the software layer that we need to see innovation to make the leap, Web Services are the key technology, providing us with a standard architecture with to solve the problem of *connecting People, Process, Information and organising the Relationships between them."*

Peter Bell, Microsoft .NET Developer Group

One question we still get asked at CBDI, is 'well what exactly is .NET then'? I think Microsoft's phrase "to reduce the overall complexity" is part of the confusion. J2EE provides similar capabilities but its features are visible in the lines of code you have to write to exploit it. Like the Lloyds building in London with its pipes on the outside one can see what it does and how it connects your system together. In contrast, .NET takes that complexity and pushes it down into the duct work of the technology platform. What you need to do to deploy a web service? Add an attribute and save the C# file; to consume one, just add a web reference.

Architecture – Inside and Out

We asked Microsoft to comment specifically on what guidance they provide for system architects when embarking on a web services project. The answer, a URL, did highlight the fact that Microsoft has really woken up to the importance of architecture and patterns.

The primary source for architectural guidance from Microsoft is from the .NET Architecture Center[21], provided by their Patterns and Practices group.

We were particularly impressed by the new Patterns and Practices area which discusses the importance of patterns and catalogues some of the more important ones. For example, go to http://msdn.microsoft.com/practices/ where you will find many of the 'Gang of Four' patterns described and implemented in .NET.

Microsoft have also published a 166 page recommendation on how to architect a .NET solution, the abstract is as follows:

"This guide provides design-level guidance for the architecture and design of .NET Framework applications and services built on Windows 2000 and version 1.0 of the .NET Framework. It focuses on partitioning application functionality into components, walks through their key design characteristics, explains how security, management and communication apply to each layer, and provides information on how the components should be deployed"[22]

The whole-hearted support for architecture and patterns is reflected not just in the support they now provide on the MSDN but also in the way they build product. Working closely with IBM on SOAP standards (GXA) has created a modular architecture on which their .NET platform will be based. A recent release of WSE (Web Service Extensions) neatly demonstrated the plug-and-play SOAP architecture. This simple add-on to .NET added encryption, identity tokens and routing to web services. Likewise the forthcoming 'Jupiter' release of the e-commerce platform will unify a collection of server products into a modular architecture for e-business.

Service Oriented Architecture

Adoption of SOA is a long-term goal for many IT strategists – Web Services being just one technology component needed to make it happen. SOA is important for enterprise IT because it provides the framework that unites the business model with the applications that provide the functionality required for efficient business. Without SOA IT systems become a disjointed collection of packages, functions and screens that consume ever-increasing resources to maintain and evolve. SOA imposes a direct correlation between business operations and software services, making it a simple task to maintain and re-factor new systems from existing services.

Microsoft has placed themselves at the leading edge of the Web Services curve with .NET and are responsible for much of the work on WS-Security and WS-Routing. In 2002, the majority of

---

21 http://msdn.microsoft.com/architecture/

22 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp

Web Services activity has been internal we asked Microsoft what advice they would give to their customers considering external Web Services:

> *"We would encourage our customers to look to understand which business opportunities can be realised or optimised as a result of Web Services. In the simplest of cases this can simply be that the opportunity cost is lowered to a point it makes sense to enter a new market. In a more complex scenario it may be that the technology creates an entirely new opportunity. The latter is less frequent and harder to find, but the former example affects every business- doing more with less.*
>
> *2003 should see the completion of the next set of core standards, providing a much needed common approach to security, reliable messaging and transactions. This will have an exponential effect on the number of business contexts where Web Services can be used. The relative maturity of the development platforms will also naturally bring in those customers who do not wish to be early adopters of technology. Lastly the success stories of 2002 will be a motivator that will move us towards critical mass. In 2002 Microsoft published over 200 customer case studies where the .NET framework was the development platform, the majority of those using Web Services as part of the solution. "*
>
> Peter Bell, Microsoft .NET Developer Group

The Mission to Jupiter

Jupiter is Microsoft's project name for their next major initiative for joined up business. It takes the various e-commerce servers and unifies the architecture on the .NET platform, taking advantage of the emerging XML standards for process management (BPEL) and the GXA SOAP stack. The first phase is due for release later this year so we have little concrete product information to share. However the following, taken from their press releases indicate a clear direction.

The first set of technologies is scheduled to be delivered in the second half of 2003 and will contain the services for Process automation, Workflow, Integration technologies, BPEL support, Integrated developer experience.

The second set of technologies is scheduled to be delivered in the first half of 2004 and will include all of the previous capabilities, with the addition of services for Content management; Commerce services; Catalog management; Campaign management; Site management; Site analytics; Targeting; Personalization; Integrated information worker experience

The Roadmap to SOA

In the transition to SOA corporations inevitably have to deal with a legacy of point solutions built on a variety of platforms. Choosing .NET as the strategic platform on which to build SOA is a first step. An architecture that uses adapters to plug in existing functionality will provide flexibility for the future migration to a less diverse set of platforms and also provide a mechanism to provide a uniform management and security infrastructure based on the GXA stack.

Migration of existing functionality, whether it is on the Microsoft or another platform, is best done using Web Services technology. We have drawn up a simple table which can be a starting point in the planning process:

Existing Functionality	Tools	Comments
VB COM. COM+, C++ objects	Visual Studio .NET or TlbImp.exe	Using the .NET-COM bridge capability allows COM objects to be wrappered within a .NET class. A runtime callable wrapper can be generated using the import utility or by adding a reference to a Visual Studio project
Java Classes/EJB	BizTalk Adapter or AXIS and .NET Studio	Using the Apache AXIS server, Java methods can be exposed as web services and the WSDL used to add a web reference to your .NET project
Mainframe Applications	BizTalk plus adapter	Platform service adapters provide Web Service interface
CORBA	BizTalk Adapters from Actional/iWay	Platform service adapters provide Web Service interface
Packaged Applications, ERP, SAP etc	BizTalk Adapters from Microsoft et al	A range of 3rd party adapters allow BizTalk to create SOAP interfaces to your ERP application
Packaged Applications CRM, Siebel, etc	BizTalk Adapters from Siebel et al	A range of 3rd party adapters allow BizTalk to create SOAP interfaces to your CRM application

Table 26 – Migration of Existing Functionality

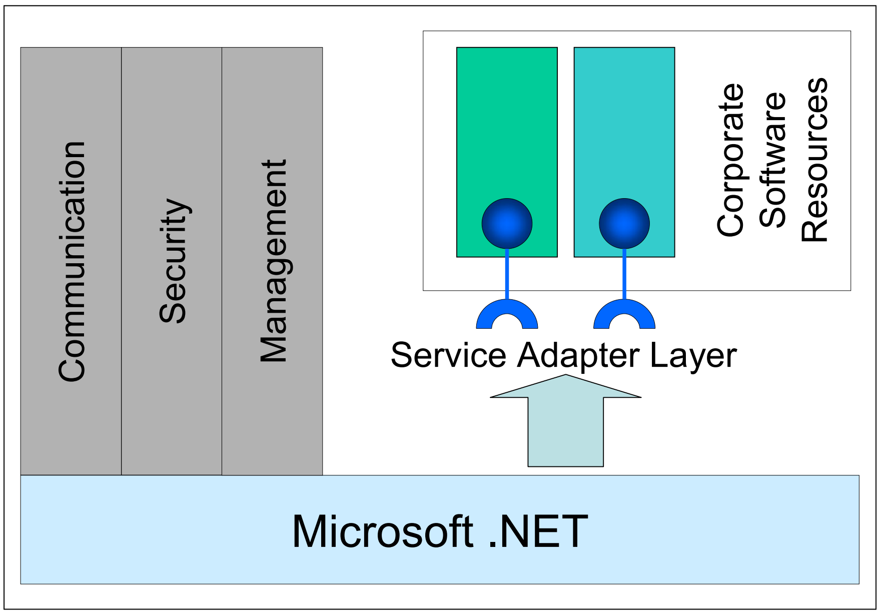


Figure 18 - Service Adaptor Layer provides access to Legacy Systems

Mobility and Smart Devices

Mobile smart devices such as phones with colour browsers, PDAs, tablet and hand-held PCs will increasingly find application in the connected workplace. The .NET platform has now found full support for these devices in the Studio 2003 release. Developers first into this area of application development had to rely on embedded C++ and VB toolkits and we welcome the unification of the compact and mainstream platform that .NET brings for 2003.

Any device worthy of the label 'smart' will have some sort of browser and the .NET server now has the ability to adapt the mark-up language to fit the targeted screen. Microsoft Mobile Internet Toolkit (MMIT) allows you to write one .NET application that will render itself on the target browser using the appropriate subset of HTML, WML for phones, cHTML for PDAs.

More powerful clients with a disconnected capability or a richer user interface are written using the SDE (Smart Device Extensions) to .NET and this too is supported from the .NET Studio 2003 release. This approach dovetails with the SOA, Web Services being provided from the corporate server to synchronise information with the mobile workforce.

Summary

We have seen a fundamental shift in Microsoft's support for concepts of software architecture and patterns. They now have a comprehensive web site for architects, complete with example implementation code. The concepts of Service Oriented Architecture now permeates the .NET platform and we can see clear differentiation between .NET and J2EE, in the way .NET increasingly hides complexity and improves productivity by doing more with less code.

Microsoft clearly sees the SOA extending out from the corporate workplace to mobile and smart devices and this will be an exciting area to be in over the next few years. The Java world has made some early wins on the PDA platform but there is a significant market developing for Microsoft here.

Links

Microsoft .NET	http://www.microsoft.com/net/
Microsoft .NET Architecture Center	http://msdn.microsoft.com/architecture/
Microsoft Web Services Developer Center	http://msdn.microsoft.com/webservices/

Practical Support for Separate Supplier and Consumer Activity

Abstract: Select Business Solutions has promoted component and service based approaches to software analysis and development for many years based on the formal separation of supply, manage and consume activities. In this report we highlight how Select is guiding its customers to adopt service orientation.

Introduction

Select has promoted component and service based approaches to software analysis and development for many years. They first articulated guidance to their customers in the Select Perspective method, published in 1998.

The original Select Perspective was the first clear articulation of how to develop component-based systems for business enterprises. Recently a new and radically revised Perspective has been published, representing a considerable advance on earlier thinking and providing a maturity of guidance that is clearly based on deep and extensive practical experience. Five years ago, software industry leaders were focused on techniques and development processes, with a strong emphasis on forward engineering. Today the new Perspective provides detailed advice on how to achieve the real business benefits that come from a mature software delivery process that is predicated on service, component and asset management and reuse, and the managed collaboration of the separate activities involved. Select's CBD experience maps directly to SOA, and shows the real world experience contained with Select Perspective. See Select Business Solutions later for an overview of the products.

Supply Manage and Consume (SMaC)

Select's guidance and products therefore, are based on separation of the activities of service and component supplier and consumer. Supply, manage and consume (SMaC) is the core philosophy underlying the Select process and toolsets, that formally separates concerns and activities implicit in distributed design by contract. Suppliers of components and services work to meet the specifications that describe the consumers' needs for the delivery of business solutions.

Supply	Manage	Consume
In principle, services may be provided by any party on any platform. Reuse of available services is preferred to the construction of new services. High levels of reliability and availability may indicate a demand for multiple supply	Communication between the parties, publication of services, and quality control are overseen by the Management function whose core objective is to enhance the value of the software assets being managed. Management of services implies matching and binding available services to multiple requirements.	In principle, services may be consumed by any party in any context on any platform. Promoting reuse may mean making the service available to a greater range of users.

Table 27: Supply / Manage / Consume

In its original and simplest form, Design by Contract incorporates the idea that components and services can be specified in terms of a logical or functional contract. While the term "Web Service contract" is sometimes used in the very narrow sense of the syntax/signature of the service interface, as expressed in WSDL/XML, Design by Contract has a broader concept of contract, which explicitly includes the semantics of the service call in terms of preconditions, postconditions and invariants. Dependencies between services (which may be covered in BPEL) are represented by the pairing of preconditions and postconditions, though very loose coupling should minimize these dependencies.

In a distributed service-oriented environment, this notion of contract needs to be extended still further to cover the quality of service and commercial issues, as shown below. The W3C WS Architecture Stack has layers for SLA, and BLA (Business), which cover these types of elements.

- Service Level Agreement (SLA)
- Quality of Service (QoS) specification
- Web services to be consumed
- Costs (time/usage based)
- Security impositions
- Web Service definitions
- other details such as client's locale, available web service mirrors, etc

SMaC Relationship Contractual Elements

The full requirements of a service contract are therefore as shown in Figure 19.

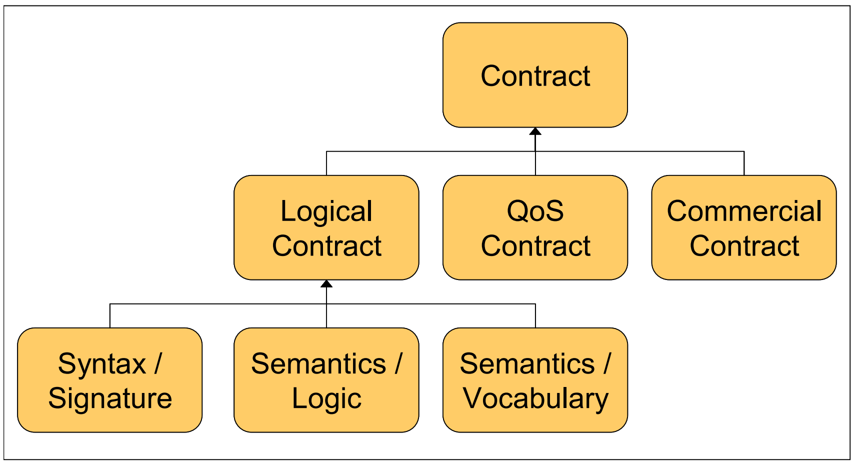


Figure 19 - Full Service Contract

Four Routes to SOA

Select Business Solutions estimates that fewer than 20% of their SOA customers have a CBD background. Many come straight from OO, pre-OO and legacy integration.

The technologies associated with Web Services and SOA have greatly reduced the costs and difficulties of at least some aspects of interconnecting complex systems from disparate pieces, as well as the perceived risk. This leads to a significant expansion in the range of applications and projects that are economically and technically feasible. In other words, while OO/CBD delivered significant benefits for some requirements in some organizations, WS/SOA is expected to be cost-effective for a greater range of requirements in a larger number of organizations.

Transition to SOA from …	… New Solution	… OO & Micro-Componentry	… Enterprise CBD	… Legacy & COTS
Starting Point	New application with little or no need to integrate legacy functionality.	Small-grain objects and components migrating to SOA	Enterprise process for large-grain reuse	Integrate legacy & COTS into a Service Oriented Architecture
Example	Large UK Retail Organisation	ABN Amro	Large US Utilities Company Large UK Utilities Company	Limit Underwriting Large UK Independent Bank
Typical Challenges	New technology Reuse, gaining skills use of tools, adoption of process	twin-track process, SMaC, organisation change, standards for services	Interoper-ability, asset management issues	service mining service integration rolling replacement of legacy technology interoper-ability

Table 28: Four Routes to SOA

New Solution

While many large IT organizations have some teams with OO or CBD experience, these methods often have not been rolled out fully across the organization. So there are still many projects and application areas that have not yet been touched by OO or CBD.

WS/SOA is in some ways easier to adopt than OO/CBD. The coarser grained nature of a Web Service brings higher business visibility and understanding. From a technological point of view, the standardization of the WS protocols makes some of the interfacing and connection issues simpler to manage. For many organisations, XML and XML Messaging, e.g. SOAP is their first step, with the adoption of the web services protocols as an incremental next stage.

Thus a direct route from pre-OO into Web Services and SOA proves a perfectly viable and reasonable one; and Select has considerable experience helping organisations to move straight to SOA. Let's review New Solution adoption using examples from Select's customer experience.

Customer	Large UK Retail Organisation
Project Name	Customer Service – responding to the queries and service requests from customers in-store
Project Objectives	Develop service oriented (Web Services) solution inline with IT strategic direction.
Initial Engagement Description	Select service team involved from the start of the project: Tool and UML training Service and Component development process training Principal consultant involvement for key milestone reviews Select associates involved on a day to day basis: Mentoring of the analysis team Mentoring and leading the technical architecture definition and implementation for SOA.
Tools Use	Select Component Architect: Business Process, Use Case and Service analysis Service Design Technical Architecture design and definition
Challenges Overcome	Risks of new technology (Microsoft .NET), new development process mitigated by tools and mentoring.
Process Issues	Coordination of projects within programme in particular the definition, implementation and use of common services, mitigated by high quality service specification, and shared models based upon asset management principles.
People Issues	Cost of the initial development effort seen as higher than a traditional approach. “Selling” and buy-in required to achieve success.
Current Status	Solution is starting to roll out to stores across the UK

Table 29: New Solution Adoption

OO & Micro-Componentry

For many organizations, CBD is perceived as merely an extension to traditional OO methods. In so-called BottomUp CBD methods, component requirements are identified using class modelling and then these requirements are bundled into physical components. Many organizations use a single track process for objects and components, such as RUP – and while this approach is certainly viable for small projects, the lack of clear separation between the functions of Supply, Manage and Consume (“SMaC”) means that it cannot scale up for large organizations.

Such processes also limit the scope for reuse of components because they take a view that components are primarily deployment-time artefacts rather than first-class tools for analysis and design. In consequence, the service architecture is considered only very late in the life of a project and opportunities for reuse at design-time and to reuse by adjusting and extending existing assets are missed. ROI from such single-track, object-based methods are likely to be considerably reduced.

In preparing its customers for the transition to SOA, Select has found that BottomUp CBD experience, whilst widely regarded as a form of component based development, in reality provides a similar appreciation of the issues to OO experience.

For one of Select's customers a UK Systems Integrator/Software Development House, a key goal in moving to SOA was to establish the SMaC framework.

Customer Name	UK Systems Integrator/Development House
Project Name	Re-architect existing solution
Project Type	OO to Components
Project Objectives	To encapsulate middle-tier business implementation from client and database concerns; to enforce a logical 3-tier model on the design of the business architecture; to deliver components for future reuse.
Initial Engagement Description	Select services team involved from start of the project: Tool supply and UML training Service and Component Based Development process training
Tools Use	Select Component Architect: Use Case and Service identification and definition. Design of XML messages Service design Select Component Manager: Cataloguing of Services to enable reuse by multiple channels
Migration/Evolution Steps	Re-factor existing business layer; Abstract business layer elements from data and presentation tiers; Chunk the business tier into components and identify service operations to be provided by each.
Challenges Overcome	Correctly understanding the implications of a service-oriented architecture; correctly encapsulate components to embody loose coupling.
Process Issues	Required the organisation to move to a Supply Manage and Consume process. Supply of services Management and Reuse of services Consumption of service within the channels Roles not familiar within the organisation in particular the management of the resultant service asset.
People Issues	ROI is medium to long term therefore management buy in is key to the success.
Current Status	Application has been delivered, but business architecture is still not fully componentised with consequent compromises in terms of coupling, quality and maintainability.

Table 30 OO and Micro-Componentry

Case Studies in Enterprise CBD

Many large organizations have found that software reuse requires more than bottom-up CBD. Enterprise CBD refers to the use of CBD within a defined enterprise process for managed reuse, typically involving a twin-track process with project/organization separation between Supply and Consume. This is sometimes called TopDown CBD.

An organization with previous experience with enterprise CBD will find some of the principles of SOA very familiar – especially the separation of concerns, the formality of contracts and communications between suppliers and consumers, and the traceability of services and business components from requirements to implementation.

Typically, Select's customers that have Enterprise CBD initiatives are looking to adopt SOA as an additional façade to their current component-centric view. The service becomes the granular, business process-level protocol for collecting together business components for provisioning to applications/solutions. As such, you could assume that this is "just another component access protocol". This would be a short-sighted approach as migrating components to SOA without analysis, does not reap the additional benefits of granularity re-factoring. Services implement a business process, business components collaborate to implement a business process, and therefore the services need to access multiple components.

Table 31 and Table 32 illustrate two existing examples of Enterprise CBD migration to SOA.

Legacy & COTS

One of the key advantages of SOA is the ability to create services interfaces to integrate with legacy systems and commercial off-the-shelf software (COTS). In the past, project groups focused on COTS implementation would often be excluded from OO or CBD methods – but these groups can now be brought inside an SOA process.

Limit Underwriting has gained considerable experience in wrapping legacy COBOL code using proprietary middle-ware technology and exposing Microsoft COM interfaces. Select tools are used extensively to catalogue the available services resulting from wrapping exercises and to model the consumption of these services into newly deployed business solutions. The result has been the speedy deployment of solutions, based on modern user interfaces, whilst preserving and enhancing the value of the legacy code.

To support future technological change, Select have recommended the adoption of the ideas of loose coupling. The provision of web services from the existing technology base is an incremental step, supporting genuine service reuse through multiple business and technological channels. A key risk identified by Limit is their dependence on obsolete technology to access functionality that has not yet been replaced by wrapped code and to manage the access to data. However the loosely coupled Web Services architecture does provide implementation transparency, and enables a "plug-and-play" approach to the progressive upgrade of functionality in an existing environment and, ultimately, the replacement of legacy database technology by an RDBMS.

Another example is a large UK Retail Bank, shown in Table 33.

Customer Name	**Large US Utilities Company**
Project Name	IT Strategy Programme
Project Type	Enterprise CBD
Project Objectives	To enable the integration of many disparate systems. Promote loose coupling and rapid application assembly.
Initial Engagement Description	Select services team involved in the process of Service and component management, in particular: To help define a process context for Supply Manage and Consume of services across the organisation Identifying a process for Service and Component Management Provide Select Component Manager configuration and usage training to support the process
Tools Use	Select Component Manager: Provide a central repository for reuse of services and components Enable the rollout of SOA across multiple development streams.
Migration/Evolution Steps	Harvest existing services and components Identify pilot project(s): Candidate for Legacy wrapping Applications Publish services for reuse
Challenges Overcome	Cross-site working making communication across projects difficult. Adoption of service oriented process and toolset is helping establish the roles and communication across the organization. The role of Service and Component Librarian – managing the service repository may be seen as an unnecessary overhead for a traditionally project oriented company. Clearly stating objectives early in the strategy and meeting those objectives mitigates against this misconception.
Process Issues	Role out of new process requires planning and support for multiple programmes.
People Issues	Training and awareness of the broader objectives is key to the success
Current Status	Service and components starting to be reused across solutions

Table 31 - Enterprise CBD migration to SOA Example 1

Customer Name	Large UK Utilities Company
Project Name	Combined Electricity and Gas Trading System
Project Type	Enterprise CBD
Project Objectives	Provide trading functionality from previous CBD projects to facilitate Gas and Electricity Trading
Initial Engagement Description	Current successful customer with CBD, engaged at project initiation to help decide on outsource supplier, and help manage the implementation and solution assembly. Training, ongoing consulting and project resources provided for the Service specification, and Service Provisioning by the selected supplier.
Tools Use	Select Component Architect, Select Component Manager and Select Reviewer
Migration/Evolution Steps	Refined documentation of the existing component/service assets with SCM Validation of three suppliers, based upon delivery of a technical component/services as a trial Split of teams between Solution Assembly and Service Provision by Third Party Implement quality measure for Service Reuse
Challenges Overcome	Solution and Service/Component collaborative and incremental working resolved through Component Manager Service specification rigour for third party development using Component Architect and Reviewer for quality assurance Service prioritisation when dealing with Gas implementation first, then Electricity second
Process Issues	Consumer/Supplier relationship with incremental working Service specification and testing/gap analysis
People Issues	Predictability of delivery timescales, given “black box” nature of service specification, need to drill in to get quality metrics – mitigated by the use of detailed models to establish require artefacts, and therefore predictable metrics Relationship management and communication between supplier/consumer – managed by the use of a central, versioned services repository (Select Component Manager) Service prioritisation, particularly when two utility types being implemented, managed by versions of the services, and related model definitions to establish/manage incremental delivery
Current Status	First increment, Gas Trading and Operations delivered, Electricity variant scheduled for the Summer, on target

Table 32 - Enterprise CBD migration to SOA Example 2

Customer Name	Large UK Retail Bank
Project Name	New Channel Services
Project Type	Legacy/COTS
Project Objectives	To remove knowledge and dependencies of the underlying legacy systems from the multiple access channels Provide a single access point for business functionality and data
Initial Engagement Description	Select services team involved from start of the project: Tool and UML training Service and Component Based Development process training Consultancy help steer each stage of the development process, in particular – Use Case and Service analysis, Service design.
Tools Use	Select Component Architect: Business process, Use Case and Service identification and definition. Design of XML messages Service design Select Component Manager: Cataloguing of Services to enable reuse by multiple channels
Migration/Evolution Steps	Evaluate integration technology Select initial delivery channel and functionality required. Identify key legacy transactions "bang for buck" to establish early ROI and reuse Define Technical Architecture Framework for Services and Legacy Execution, including data transformations through the layers.
Challenges Overcome	A well-defined Service and Component Based Development process enabled potentially difficult synchronization of multiple delivery teams. E.g. Channel, Legacy integration, middleware.
Process Issues	Required the organisation to move to a Supply Manage and Consume process. Supply of services Management and Reuse of services Consumption of service within the channels Roles not familiar within the organisation in particular the management of the resultant service asset.
People Issues	ROI is medium to long term therefore management buy in is key to the success.
Current Status	The business channels are successfully using Middleware services. New channel projects starting to reuse middleware services.

Table 33 - Enterprise CBD migration to SOA Example 3

Select Business Solutions

Select Perspective

The new Perspective process provides detailed advice on how to achieve the real business benefits that come from a mature software delivery process that is predicated on the managed collaboration of the separate activities involved. Select's guidance and products are based on separation of activity into the activities of supply, manage and consume (SMaC).

The SMaC process is inherently service oriented. The process is predicated on reusing existing service and component assets where they already exist, either directly or by extension, and provides a formal framework for distributed design by contract and reuse from third parties. The process together with tools orchestrates the communications and work of suppliers and consumers.

Select Toolset

Select Component Factory is a suite of UML based modelling tools that actively support the SMaC framework. Analysis and design activities are supported by Select Component Architect, which adds Business Process Modelling and Data Modelling notations to UML. The Component Factory covers modelling from understanding the business context to the design and implementation of the physical database. Software solution designs are expressed in terms of the protocols of service operations.

Select Component Manager supports component and service asset management, including publishing. Select Process Director provides support for the definition, customization and deployment of the software development process based on the Select Perspective.

Select Support

Select Professional Services provide advice and guidance and help to manage organisations through every step of the adoption of Service Oriented Architecture and the development infrastructure to support it. Focused on skills transfer, services include training and mentoring of the initial project teams, through to planning and guiding the implementation of organisation changes.

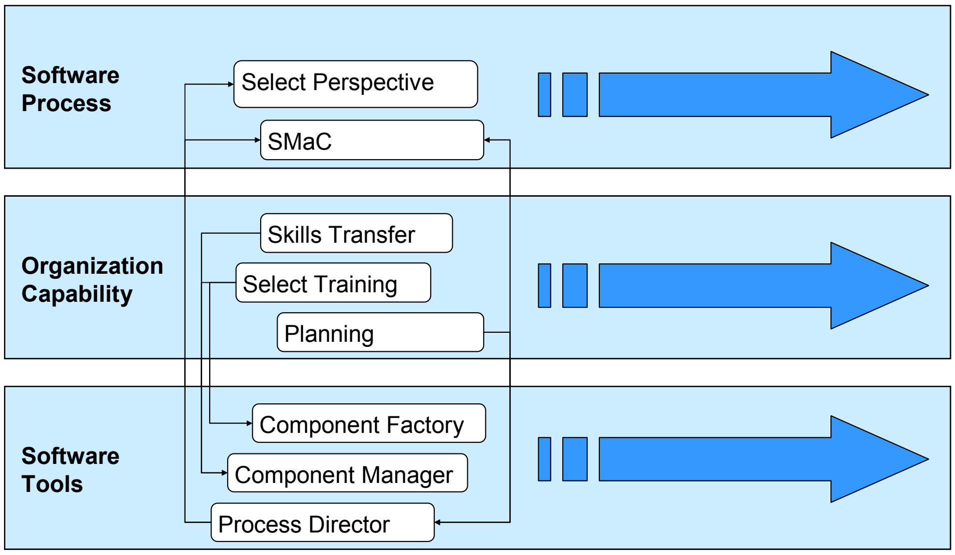


Figure 20 - Select Business Solutions

Links

Select Business Solutions: http://www.selectbs.com/

Web Services Roadmap

A CBDI Report Series – Guiding the Transition to Web Services

Web Services will shortly become a core infrastructural technology, and potentially open up many new opportunities at both business and technical levels. The key questions for enterprises are where to start, what to do now, and how to prepare the ground for a period of rapid and constant evolution while maintaining high quality and minimum risk? The CBDI Web Services Roadmap provides a framework for planning and managing the introduction of Web Services and SOA, together with guidance on many related topics.

CBDI Web Services Roadmap deliverables are available without registration at

www.roadmap.cbdiforum.com

Subscribe to CBDI Forum

CBDI provides continuous guidance and best practices for the Service Oriented Enterprise. In addition to the monthly CBDI Journal and other reports, subscription includes access to workshop materials.

For more details and to subscribe see:
www.cbdiforum.com

About Us

We aim to provide unique insight on component and web service technologies and processes for end-user organizations and vendors. We provide high quality analysis and other information resources on best practice in business software creation, reuse and management, with the highest level of independence. Our competencies cover business issues through technical platform and application design.

Learn from CBDI

CBDI provides a number of channels including:

- Subscription services - access our continuous commentary and practice guidance in the monthly CBDI journal, and via workshop materials.
- Workshops and seminars - jump start your team with in depth briefing and guidance on advanced architectures, processes and practices from CBDI analysts.
- Consulting - put CBDI's insight to use in your organization. We can assist in many ways including application audit and guidance, product management and marketing advice, technical and business evaluation for investment

Who uses CBDI Resources?

Technical leaders including architects, business analysts, CIO's, consultants, CTO's, lead developers, development managers, process engineers, product managers, project managers, researchers, strategists, technologists and technical managers. Membership is split 40% USA, 50% Europe.

Contact us:

For further information on any of our services contact us at:
info@cbdiforum.com or +353 2838073 (Ireland)

CBDI