February 2002

interact

CBDi forum

The Journal of Web Service & Component Based Business

When and Where Shall We Use Web Services?

Editorial Abstract: An important question for everyone right now is when and where to use web services? For the software industry the question is a survival issue. For end user organizations it is more about assessing when the broader technology will be fit for purpose, and balancing opportunity and risk. In this month's INTERACT our reports address this question from a number of perspectives.

Last year was a year like no other in the history of the software industry. It could easily be summed up in the Latin phrase used by Queen Elizabeth to describe her royal family's most difficult year as the software industry's version of "annus horribilis." Its initial phase was marked by financial turmoil, and just when many were forecasting that the worst was over, the terrible events of September 11th plunged industry as a whole into unforeseen depths of despair.

Right now there is considerable disagreement on the timing of the financial recovery. Whilst there are signs of spring on Wall Street, with a smattering of better results, industry leaders such as Bill Gates and leading economists are predicting that this year, particularly for the software industry will remain at best, flat. And this also reflects the diversity of activity, with some services companies doing quite well, whilst others are literally struggling to survive. Security and trust software providers are experiencing a boom period, whilst many tools and platform vendors are wondering how long they can hold out. So its no surprise that the software industry is viewing web services as its salvation. No surprise that vendors are looking around for new ideas that will drive new projects and capital investment.

But like most new technology trends web services will go through a natural evolutionary process, which as sure as night follows day, means that following technology excitement, there will be a period of fear, uncertainty and doubt, (FUD) followed by small successes that lead to broader confidence and general acceptance. And today we might say that we are right slap bang in the middle of the FUD phase. Whilst there is great excitement about the technology,

continues...

inside

everyone is finding that the realities of a managed web service environment are much more complex than slapping a SOAP interface on the top of an existing component. There are significant issues to be resolved relating to managing the environment outside the component container, managing endpoints, security and trust and delivering reliable service and many more. It's rather like the exciting time back ten years ago, just after we all discovered client/server, and were enthusiastically delivering systems that were impossible to deploy!

I am not suggesting for a minute that web services will go the same way as client/server, but that there are a whole raft of issues that need to be resolved. In many cases these actually can be resolved by more comprehensive platform and tools products, and the almost feverish levels of activity in the tools market particularly gives me confidence that these problems will be resolved.

In this month's INTERACT we look at some of these issues. We commence by looking at "just how do you identify what services you need?" You might be forgiven for thinking that web services are a technology looking for a business problem to solve. But in our report **Identifying Web Services**, we provide a framework for thinking about how to address business problems more effectively by using the web service based architectures. In our report on **Consuming Web Services** we continue our multi part report on the Business Service Server conceptual layer, the new platform and tools capabilities that are rapidly filling the gaps referred to earlier, to provide management, collaboration, aggregation, assembly and consumption functionality.

Though everyone is hoping that Web Services are the salvation, for many IT industry players they may in fact just be the final nail in their coffin. Taken to the ultimate conclusion, who will buy your hardware and software if everything is now consumed as a service? Best make sure that the service providers and hosts are using your products, as there may be scant other opportunities. It will be no surprise to many that IBM and Microsoft have been at the heart of the web services initiative. These two companies have invested massive resources to conceptualize and then productize web services standards, platforms and products. In our third report this month **IBM Seeks Partners to Drive Adoption of XML Web Services** we look at the work that IBM is doing to encourage their customers to embrace web services as practical solutions to today's business problems. IBM is firing up a strong partnership program in their attempts to ensure that Web Services are 'Powered by WebSphere' which we report on.

In our final report **Xara Online Connectable Modules**, we look at a successful ISV and report on how they are transitioning their products to the web services world. Xara a UK software company is looking move beyond the traditional software sales approach by transitioning to a "pay as you go" model. We report on their "Connectable Modules" that combines hosted development and execution with Web Services.

It is clear from our analysis that the web services environment is undergoing massive change, that new capabilities are emerging almost by the day, and that the market and technology are really quite unstable. However the upside is that with caution, and for the right applications, the capabilities are there right now. The challenge for everyone is therefore to assess the balance between business benefit and technology risk. As we have said before the raw technology of SOAP etc is no technological risk whatsoever, it's extremely simple. The risks are more about the management and trust. This leads us to the conclusion that it is essential to select the right applications in the short term that balance the amount of DIY with business benefit. It is likely that the vendors will be disappointed with the level of real enterprise level investment and project activity this year, as there is too much movement on all fronts. However industry players who have not started to transition their products are already running extremely late. For end user organizations, now is the time to run those pilot, narrow path and groundbreaker projects or whatever you call them, and get ready for the real action in 2003.

David Sprott david.sprott@cbdiforum.com

Identifying
Web Services

By Richard Veryard

In a couple of recent articles for Interact, we reviewed techniques for the identification of components. In this article, we are going to look at techniques for the identification of web services. We suggest that service identification is not a linear progression of conventional analysis techniques, and that services require consideration of a number of additional perspectives.

Background

Although some of the revolutionary aspects of web services have been widely discussed, in the CBDi Forum and elsewhere, the design of web services is often presented as a continuous evolution - following the same basic engineering principles as software artifacts in general and software components in particular. For the developer of web services, the technological novelty is situated mainly in new platforms, protocols and description languages.

While these technical aspects of web services are undoubtedly important, this evolutionary view overlooks some important challenges facing the web service developer. What is the right scoping and granularity for a collection of web services? How can a web service be given a clear and meaningful identity, so that it can reach its target consumers most effectively?

With component-based development, the concept of twin-track development has become commonplace. Service-based development takes this concept further; we may typically expect the responsibility for the service to belong in a separately managed entity - whether as a separate cost center or profit center within one company, or as an independent web service provider. This means that transfer pricing or acquisition method and cost becomes a key element of web service design.

Many organizations dipping a toe in the shallow end of Web Services will not want to be bothered with such issues as transfer pricing, and we sympathize with this. However, transfer pricing introduces a degree of rigor into the management of web services - and forces attention on some issues that might otherwise be overlooked.

Design Goals

From the service provider's perspective, we may assume three generic design goals for web services.

1. Each service should offer a meaningful and attractive proposition to users. If we assume that users have a free choice whether to use this service or not, then this becomes essentially an exercise in marketing. Marketing people commonly use a 4P model: Product, Price, Place and Promotion. We can adapt this to a 5P model for web services, as shown in Table 1.
2. The collection of available services should form a coherent whole. This demands attention to holistic system properties such as performance and security, as well as Quality of Service (QoS). Composition of services may be relevant from a marketing

continues...

5P	Relevance to web services
Product	Functional and non-functional characteristics of a single service - possibly described as a Use Case.
Price	Charging model and charging rates. Within a single enterprise, this would cover transfer pricing.
Platform	Delivery mechanism, protocols, infrastructure.
Publication	Promotion and brokerage mechanism – how potential users discover, evaluate and negotiate the service.
Packaging	Bundling or aggregation of several services (possibly from third-party sources) - for the purposes of promotion, pricing or delivery.

Table 1: 5P model applied to web services

perspective, if there is any bundling (Packaging) of services for price or promotion purposes.

3. The delivery of the services should be sustainable, across a broad range of demand scenarios. Considerable emphasis is placed on software economies of scale, commonly known as **reuse**. (If any reuse is achieved by the service provider, this is only of indirect interest to the service consumer, because it may affect some characteristics of the service such as its price or reliability.)

The key issue for the identification of web services is that a purely functional design approach, described in UML and based on use cases, only addresses the first of these three design goals. UML supports a computational model of a system, but does not adequately support the other four viewpoints of RM-ODP.

Even for the first design goal, UML alone may not be sufficient. Some aspects of specification require the information viewpoint of RM-ODP, which can be expressed in XML or RDF; among other things, this is necessary to specify context-sensitive services (for example, where key parameters are inferred from an XML or RDF schema).

For the other two design goals, UML leaves a lot to be desired. One possible modeling solution has been proposed, which slightly deviates from pure UML - and is not supported by UML tools. Given that use cases are used to describe services, a form of use case package is then used to "bundle" logically related services together. At this point the use case package itself could be described by explicitly defining package ("holistic") properties, which would be implicitly defined for all of the encapsulated "service" use cases.

The designer needs to pay attention to issues of cohesion and coupling between web services - but not just from a computational viewpoint. Whereas component identification can be regarded as a simple clustering exercise, service identification needs to be regarded as a more complex clustering exercise - because it relates to the commercial and technical viability of the service provider as well as the functional integrity of the service consumer.

Top-down and bottom-up

Service identification needs to address service provision at (at least) two levels - Sun Microsystems refers to these two levels as **macro services** and **micro services**, which seems to be a useful nomenclature. There is generally a hierarchy of services where micro services are the operations offered by the components and the macro services are the business/user facing services.

Sun's thinking in this area seems to be grounded in the history of UNIX, where small utilities could be piped together to form larger wholes. Strict adherence to the UNIX legacy might suggest a particular architectural style (pipes and filters), with potential implications on other architectural requirements. However, the general macro/micro hierarchy does not entail the pipes and filters style.

This macro/micro hierarchy then implies two design approaches: top-down and bottom-up. Top-down service identification starts with the macro services, and defines a set of matching micro services, while bottom-up service identification creates macro services from micro services by bundling. Either way, some iteration may be required, since different design considerations apply at the two levels, and some design trade-offs are typically involved.

Service Composition

Services are combined in at least two different ways. The service provider composes micro services into macro services, while the service user composes services (micro and macro) into business processes

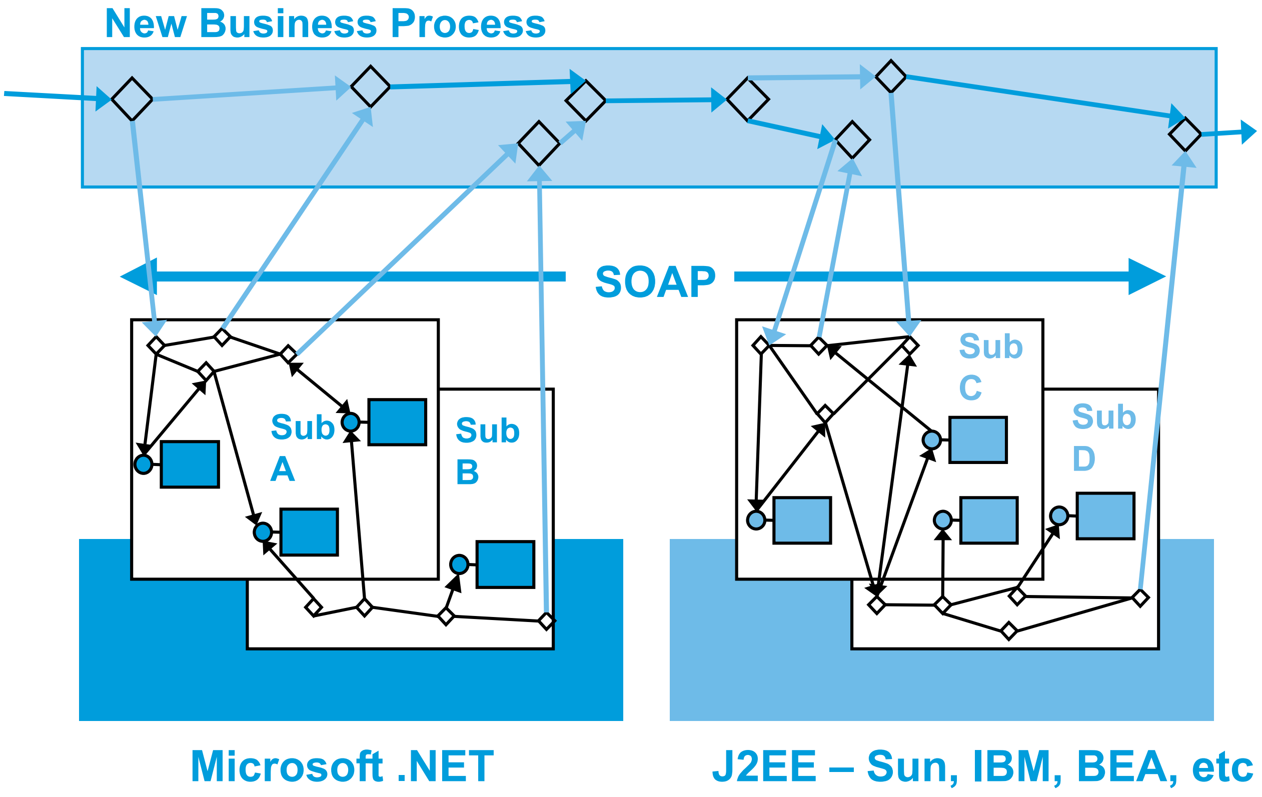


Figure 1: Macro Services: Aggregate Component Interfaces

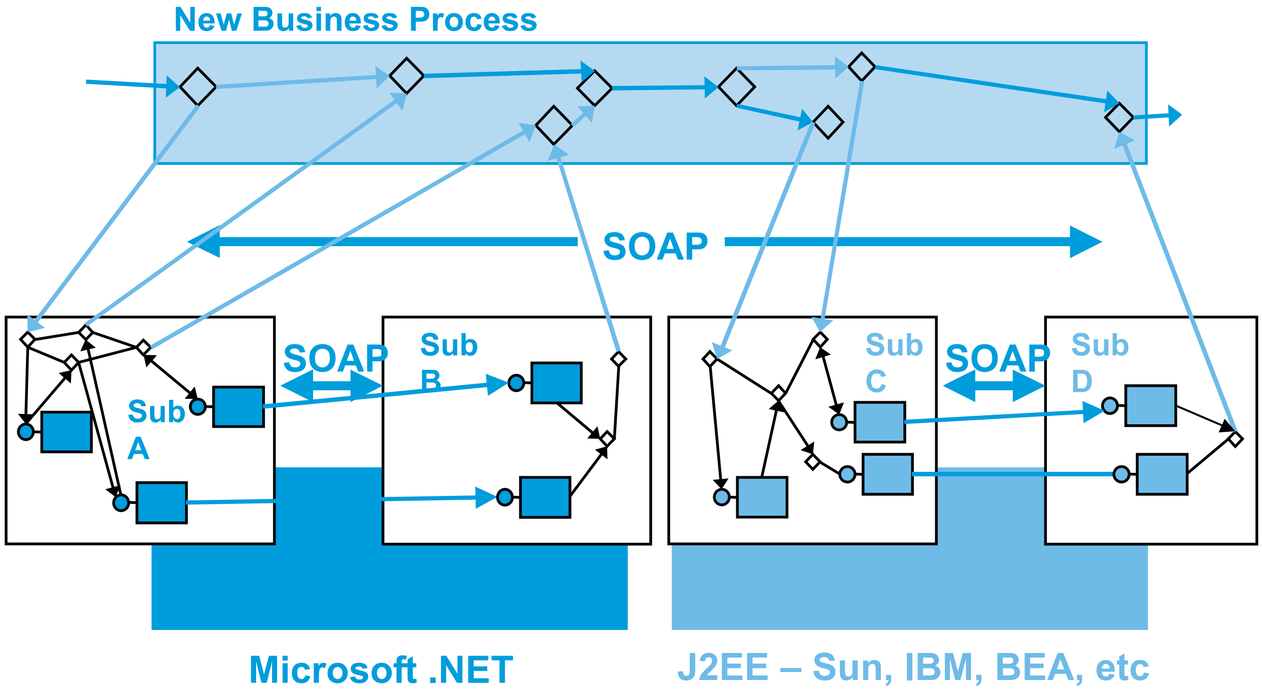


Figure 2: Micro Services: Formalize Published Subassembly and Component Interfaces

and applications. Both dimensions of composition raise quality of service (QoS) implications, and the possibility of feature interaction.

Conversational services - where more than one "round trip" is required to complete a business transaction - represent a third form of composition. This highlights the importance of intermediate mechanisms - such as data caching – to address performance and security requirements.

The added value provided by a web service may comprise some calculation or computation. More often,

continues...

it will offer access to some proprietary information. For example, the Microsoft Passport service consolidates third party data and makes it available under secure services to those that are entitled to see it.

This means that where several services are built around the same persistent data, then the identification and utilization of these services needs to be considered for the whole set, rather than one service at a time.

The granularity of web services is then a commercial decision, as much as an engineering one - it has to do with the logic and meaning of the service relationship between provider and consumer.

Aggregation and choreography of web services - including bridging between heterogeneous vocabularies - is supported by a range of web service tools such as iNsight from Iopsis[1].

Quality of Service and Future Proofing

Although it may be conceptually easier to consider Quality of Service (QoS) in relation to a single (micro) service, it is important to see QoS across a whole service platform (from the provider's perspective) or across a whole business process (from the service consumer's perspective). There are potential conflicts in terms of feature interaction and managed variation. The OMG has recently issued an RFP for the modeling of QoS within UML, but we are not confident that any work from this quarter will adequately address the holistic aspects of QoS.

Security experts make the point that service-level agreements (SLAs) need to be managed at the macro level, rather than for each individual service. As Dr D.K. Matai of m2ig software explains, this has serious security implications as well as performance implications: *"A number of separately negotiated SLAs ... can leave gaping holes in the organization's defenses as it negotiates strategic alliances."*

Within a single company, traditional software applications are designed for a given number of uses and volume of transactions. Although there is typically some latitude for expansion, significant scaling up usually requires more hardware, and often requires some additional software development.

Furthermore, traditional software is used in a predictable way, for a fixed segment of a fixed business process.

In contrast, web services may be used in unpredicted ways, at unanticipated and irregular demand volumes - even within a single company. Web services come with the promise of Plug-and-Play - even though this may alter transaction volumes by an order of magnitude.

This means that the service provider must pay considerable attention to capacity planning and resource smoothing. Resource smoothing implies either a range of services run on the same platform with different demand characteristics, or a variety of consumers with different expected demand. Differential service[2] (possibly with differential pricing) is a useful technique for achieving resource smoothing as well as increasing the economies of scale.

The service provider should be able to upgrade extremely quickly in response to demand changes, and should be able to migrate to a larger platform (or a new version of the platform) without interruption to the service (migration transparency).

Technical considerations - where are the boundaries?

One of the most frequent questions our members ask about web services is whether a particular chunk of functionality should be implemented as a web service at all. *Should this be done using XML Web Services? Do I implement this interface with SOAP, CORBA or COM? Do I implement it with SOAP or Message Queuing? I know I need to present an interface here, but what technology should I use? Should I offer the same interface in many technologies to allow consumer choice, or at least enable transition?*

Sub Assembly (like Customer)	internally uses native platform protocols (e.g. COM) to communicate between low level interfaces, but interfaces external to the sub assembly are SOAP to facilitate integration into multiple systems
Application	same as sub assembly
Platform	communications between components on same platform use native protocols; communications to different platforms use SOAP - this implies that some interfaces will be duplicated. i.e. both a native and SOAP interface doing the same thing.
Organization	external interfaces to customer, partner, etc

Table 2: Identifying boundaries and interfaces

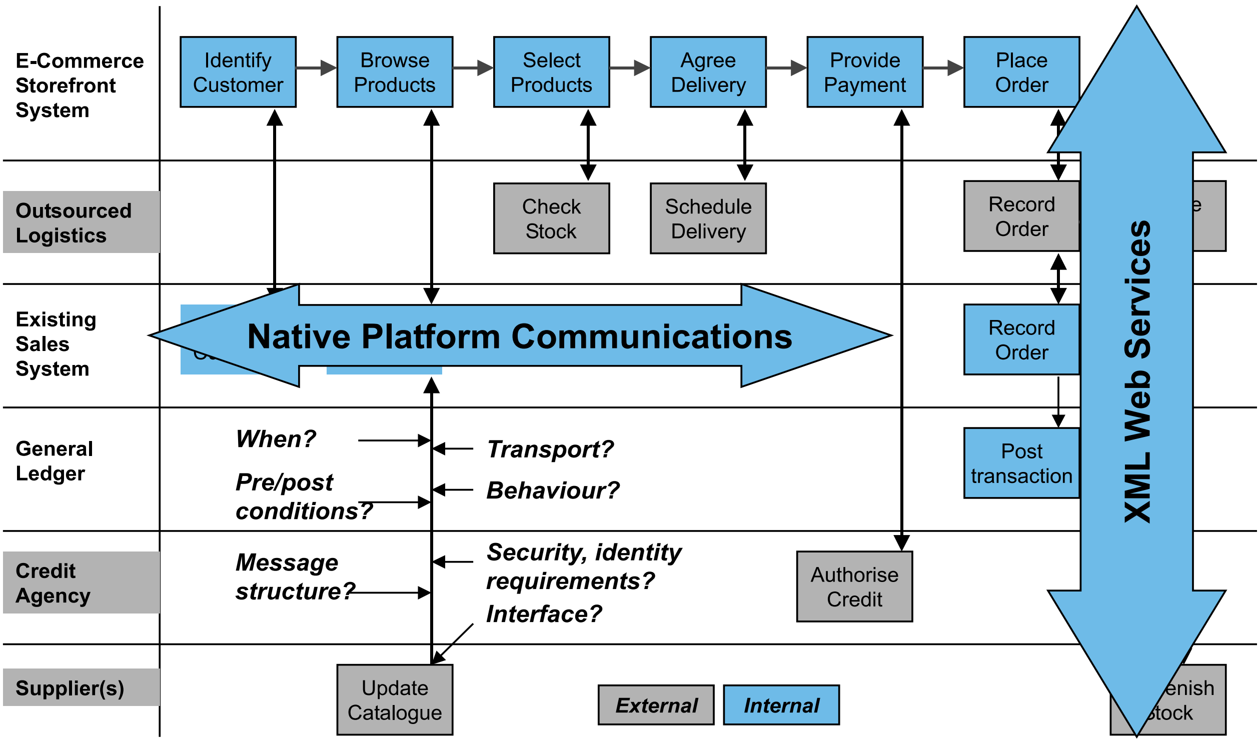


Figure 3: Web services may cross system boundaries

One of the main considerations here is the presence and stability of boundaries - where an interface crosses an identifiable boundary then it could be a candidate for SOAP. The identification of boundaries may be done from a business model - for example a process or workflow model divided into swimlanes, with organizational or system boundaries between the swimlanes. See Figure 3. Static models can be drawn in a suitable extension of UML; however, such models do not show the dynamic stability of these boundaries and are therefore not suitable for analysing or designing the boundaries themselves. Swimlane diagrams are widely used for business process modeling and workflow[3].

Service Pricing

There are several basic types of pricing scheme suitable for web services.

continues...

Flat Rate	The user pays a fixed amount, regardless of the amount of usage made of the service.
Usage-Sensitive	Users pay a fee that is partially based on the amount of usage that they make of the service. As an example of this, AT&T wireless has recently become the first US company to charge mobile internet usage based on the quantity of data downloaded. This charging model is already used successfully by AT&T's Japanese partner DoCoMo.
Transaction-Based	As with usage-sensitive pricing, the more you use the service, the more you pay., But the price is determined by the characteristics of the transaction, not by the volume of data.
Differentiated	Having several alternative prices, or a price formula, where the selection or calculation of price depends on some characteristics of the customer or transaction.
Random	Having several alternative prices, which are offered to customers at random, or on a rotating basis. This allows a company to test out demand at different price levels, and to measure the price-elasticity of demand - in other words, the degree to which customers are influenced by price rather than other factors.
Opportunistic	Ad hoc pricing - charging whatever you think you can get away with.

Table 3: Web service pricing schemes

Note that the choice of pricing scheme is logically separate from the billing mechanism and the granularity of delivery and billing (so-called MicroBilling).

Comparing pricing schemes

A pricing scheme should have the following characteristics.

transparent	meaningful and understandable by customers and suppliers
predictable	customers can predict how much the service is going to cost suppliers can predict how much revenue the service is going to yield
workable	technically feasible and efficient to administer
incentive compatible	encouraging economically efficient distribution patterns of demand and supply
stable	steady and sustainable over time, with reasonably steady price adjustments
economic	capable of being delivered at a reasonable return for the supplier
fair	level playing field - avoiding unfair discrimination between different classes of customer and usage, avoiding anti-competitive practices

Table 4: Pricing scheme requirements

In regulated industries such as telecoms, the regulators typically pay considerable attention to pricing schemes - especially fairness and transparency. For example, telecoms regulators favor such patterns as **Common Carrier**, which force prices to be published, and rule out discrimination between users with identical service requests.

Regulators are extremely wary of differentiated pricing, as they suspect it is being used to give unfair advantage to large powerful customers, or to block out competitors. However, differentiated pricing can usually be justified if it is related to different costs of service provision, or if it is intended to encourage a more even distribution of demand. For example, some services may offer off-peak rates, either to try and shift demand away from peak periods, or to try and increase utilization of resources that would otherwise lie idle.

Ideally, the interests of provider and user should coincide. Demand smoothing leads to higher efficiency and reduced cost. Users who want to use a service at the most popular times, should be willing to pay more than users who are willing to use the service at other times. In some cases, however, the provider may gain extra profits by artificially congesting the network - in other words, the interests of the provider clash with the interests of the user community. This is known as **incentive incompatibility**.

Customers like flat-rate schemes, because they know how much it's going to cost in advance, regardless of the amount of usage. Suppliers are often uncomfortable with flat-rate schemes, because excessive usage of the service could push costs up to an uneconomic rate.

Predatory pricing refers to the practice of posting economically unsustainable prices – often used as a tactic by large companies with deep pockets, to bully or break smaller competitors.

Tools

We asked a number of tool vendors to tell us how their tools supported component and service identification, and for the decomposition of business requirements into discrete components and services.

Drivers for identification

Service identification therefore has a number of important design drivers, which go beyond the design drivers for software components. We can identify three important areas here: process design, data analysis and design, and trust specification.

System architects and developers should make a conscious choice between document-driven services and process-driven services. While document-driven services may be easier to provide, process-driven services may provide a better fit to the consumer's requirements.

Process design

In talking to CBDi Forum members about the design of components and web services, we constantly hear the message that process is the place to start. Services are generally business facing. They often cause or enable processes to be altered for some business advantage (time, information currency, information availability, new/better collaboration, etc). The identification of services is a natural attribute of business design and process definition.

Data analysis and design

Last year, there was some discussion in the CBDi Forum about "new wave data analysis", Figure 4

Facility	Tools
Specification/documentation facilities, so that the designer can record and edit his design decisions.	EpioCam, Websphere
UML or other diagrams. An experienced designer can "see" the components from the diagram, perhaps using some processes/techniques described in the online help or taught on training courses.	There is a vast range of tools offering UML modeling. These include EpioCam, Paradigm Plus, Poseidon, Rational Rose, Select Enterprise, System Architect.
Description/store/search facilities, so that the designer can easily locate existing components & services.	CapeConnect, EpioCam, Select Component Manager, Websphere
Patterns provided in tool, so that the component identification may sometimes follow automatically from a particular pattern.	EpioCam, OptimalJ
A clustering algorithm is built into the tool, to perform a first-cut decomposition automatically	To date, the only tool we have identified with a clustering algorithm that may be used for this purpose is Advantage:Gen. However, Computer Associates is not currently promoting this facility.
Other.	No other relevant facilities have been identified.

Table 5: Tool support for service identification

illustrates how analysis of data ownership leads to an understanding of process opportunities and by implication requirements for documents (data exchange) and services (logic complementing the data exchange).

Service design requires both an understanding of process and an understanding of data. Within a typical development organization, there will be some analysts who prefer to start with data models, and others who prefer to start with process models. There will also often be different organizations responsible for data and process respectively. The service architecture emerges from a dialogue between these two groups and modeling activities.

continues...

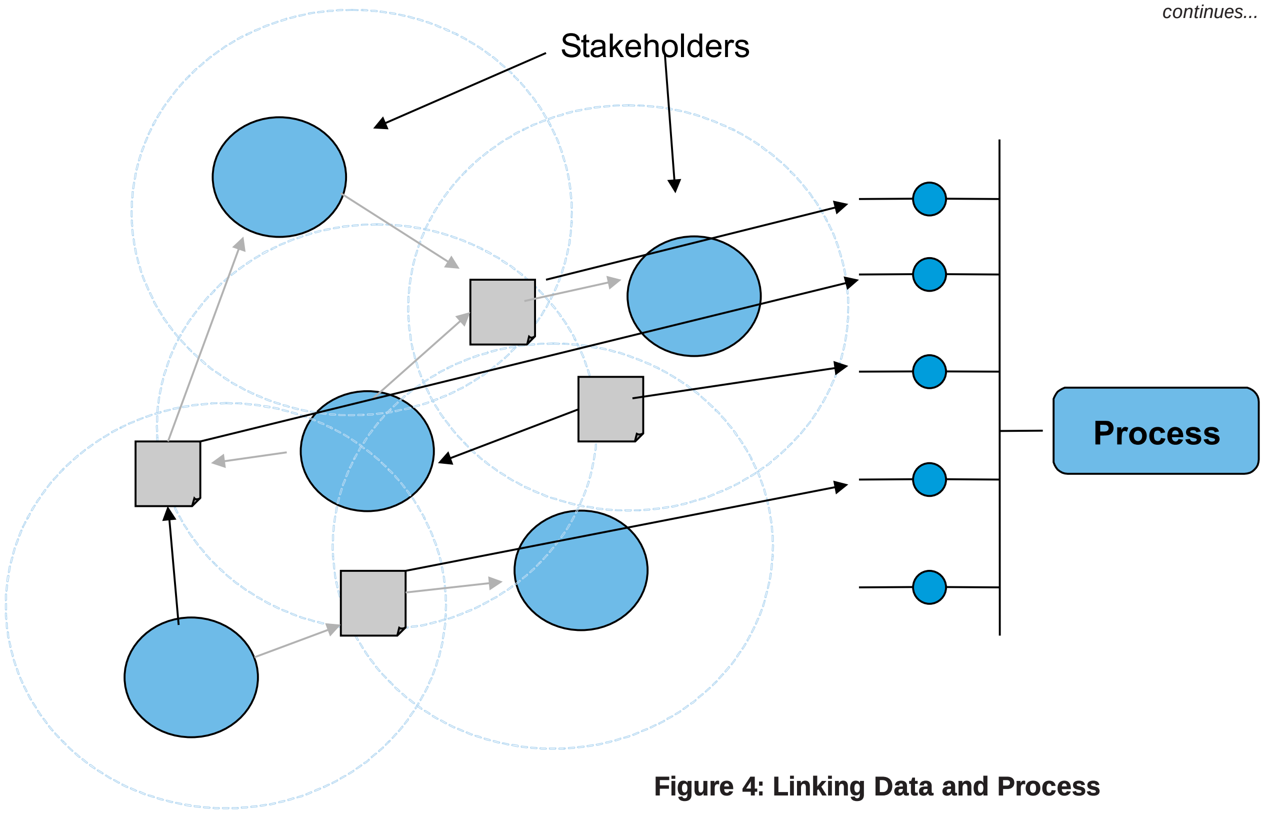


Figure 4: Linking Data and Process

Each stakeholder has a context, which can be expressed as a local data model. A distributed business process can be understood as a series of collaborations or conversations between different stakeholders - and 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.

As we've said before, we no longer need to assume a single "global" data model spanning the whole business process. This is both infeasible, unnecessary and actually counter productive. Often the stakeholder viewpoint developed in data analysis allows us to identify the various contexts that we may consider as variants to one standard process. Smart web services, as envisaged in the Sun Microsystems vision, provide the required context mappings and bindings; persistent data storage will typically be distributed across multiple platforms and formats.

Trust specification

The idea of collective or aggregate specification is highly relevant here. Consideration of integrity units provides a useful step towards identifying differing levels of horizon, necessary for different activities.

As we first encountered it, the idea of the Integrity Unit was presented as rubber-banding collections of components that would then be the unit of upgrade. In this context the integrity unit was the boundary across which the dependencies were a) minimized and b) thoroughly understood. This idea can now be extended to identify a range of additional integrity units - for example, for behavioral monitoring and or trust management, the business process, for upgrade/ replacement, for usage certification (collections of services), for environmental testing and SLA definition.

Final Remarks

In our opinion, the identification of web services has not been adequately addressed - even by the web service evangelists. The prevailing assumption seems to be that design process for web services largely replicates the component design process.

However a technical orientation of web services is missing a real opportunity to get a better and more dynamic fit between the business and software systems. This has important implications for the way that we should think about service-based development across the business/software spectrum. Further it is imperative that we identify deployment and management issues concurrent with the business and technical design tasks. In most cases there will be multiple overlapping scopes under consideration that concurrently develop the necessary infrastructure and management that meet the business requirements.

In this article, we have tried to show how some of the new aspects of web services lead to a significantly different approach to web service identification.

Richard Veryard *richard.veryard@cbdiforum.com*

1. http://www.iopsis.com/whitepapers/iNsightWhitepaper.pdf
2. Design Pattern - Differentiated Service (Fewer Interfaces than Components) CBDi Report December 2000, http://www.cbdiforum.com/secure/interact/2000-12/design_pattern.php3
3. http://www.popkin.com/services/enterprise/enterpriseware.htm

Identification Process – Understanding Service

Identify new business process

For each flow

- Timing - determine asynch/ synchronous behavior
- Determine transport mechanism
- Determine routing
- Transaction management - determine integrity constraints
- Protocol/data conversion needs for non XML Web Services
- What SLA is required?

For each process step

- Identify services required

For each piece of information

- What needs to be
  - Available real time?
  - Consistent real time?
- Who is the owner?
- What is the source

Consuming Web Services

By Jonathan Stephenson

This report builds on two previous reports on the subject of the Business Services Server. In the first we described the concept of a managed environment for Web Service consumption and provision, and the second looked into some of the solutions available today. We concentrated on the web service provider and featured IBM's hosting toolkit, aimed squarely at the web service provider who wants to charge for a service. This month the consumer's view will predominate.

Introduction

The BSS concept has provided a rich seam of ideas, insights and more surprisingly, new server products. When we started this series we had not realized that so many companies were already thinking along the same lines. Last month's report unearthed some innovative servers that we had not seen before - and yet more have been brought to our attention for this report. In particular I have added Interkeel and Altoweb to our survey and also spent some time talking to Mark Prichard, BEA's product architect to see what is in the pipeline for WebLogic.

Before we look at these platforms however, a more technical section will explain some of the 'how to' techniques to crystallize your understanding of the developer's view of XML web services.

The Developer's View

Whether you are developing web services on a Java or Windows platform, the development effort required to convert the component interface into a service is minimal. With .NET you simply decorate the method declaration with a [WebService] and include the necessary namespaces. In the COM and Java environment a wizard will usually take you through the steps necessary to create the WSDL file and copy the files needed to map the soap call to the component. There are two distinctly different options when building a web service client: one is to use a wizard to create a proxy and the other is to dynamically invoke the service at run time.

If you have used Microsoft's SOAP toolkit the soap client is initialized with the URL of the WSDL file and at run time the soap client dynamically 'becomes' the remote object. This dynamic binding has the disadvantage that when you are writing code, none of the Intelli-sense features of the coding tool can work because the object is not created until runtime. Of course there is nothing to stop you making a class to wrap the call to SOAP. The .NET framework goes much further by providing a local proxy object that provides all the type information required to popup lists of methods and parameters at design time.

In the Unix world, or more specifically when using Apache, the web services deployment steps will soon be very similar to that of .NET. The latest version of SOAP for Apache is called AXIS. It is not fully released yet but you can download it from xml.apache.org/axis/index.html. With Tomcat and Axis, turning a Java object into a web service is even easier than with .NET!

You simply change the .java extension to .jws. When the J2EE server encounters the .JWS file it fires up the AxisServlet and this in turn compiles it and handles all the SOAP calls. Now, to consume the service from a client you need only to import the ServiceClient, create the ServiceClient object and set the endpoint URL as your .JWS file. The ServiceClient now behaves in exactly the same way as the remote object, but as in the Microsoft SOAP Toolkit the object has no 'type' information until runtime.

Axis like .NET goes the extra step needed to completely hide the SOAP layer from the developer by generating a Java stub. A tool called wsdl2java.exe takes a URL to the wsdl file and builds a Java class on the client ready for you to include in your code. Once you have the local stub, the Java IDE, like .NET studio really doesn't care that the object is a remote service, and makes

continues...

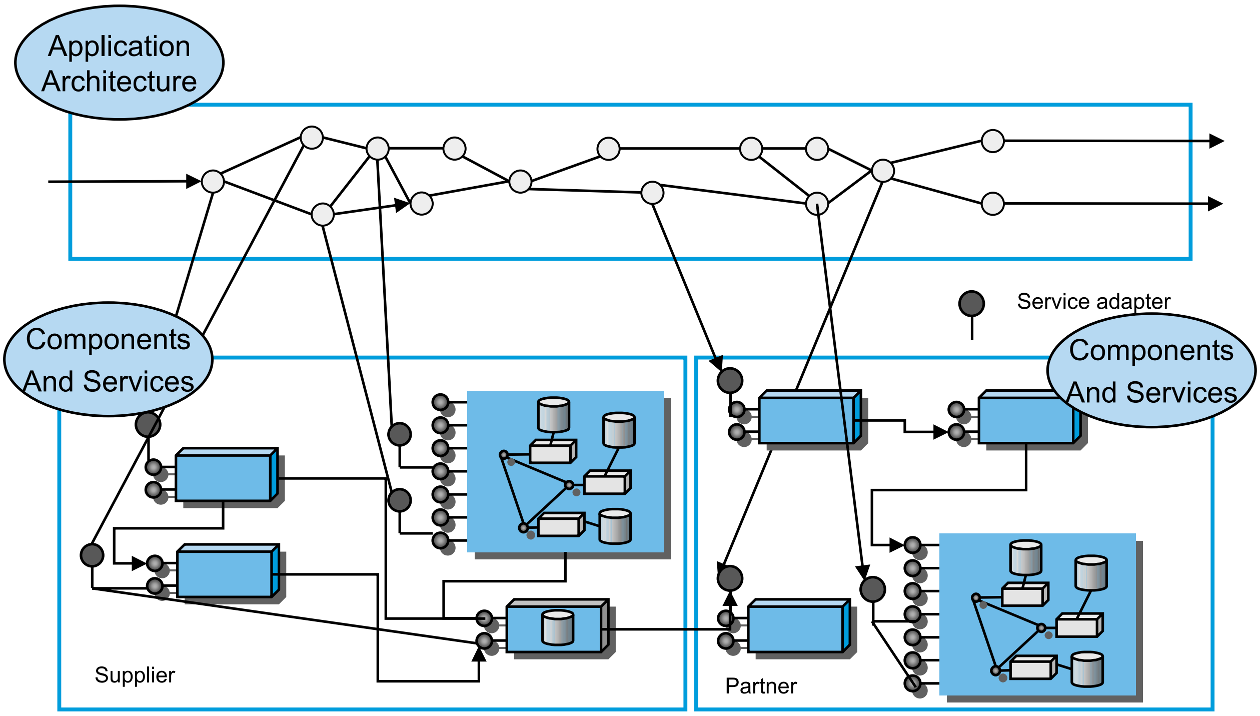


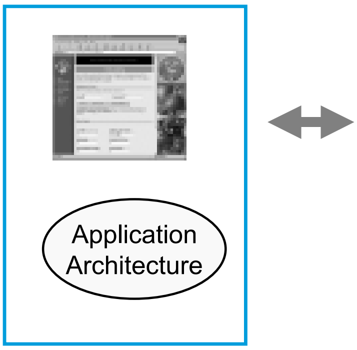
Figure 1: A Network of Dependencies

type and parameter information available in the design-time environment.

I think this very brief overview is enough to illustrate that developing client code that uses an XML Web Service is really very straightforward. Also, that whether the platform be Java, .NET or COM, the steps are starting to look remarkably similar.

The Spaghetti Web Problem

The temptation for developers will be to create a network of dependencies within the enterprise applications; fixing the various identification and QoS issues on a 'per service' basis. These problems were bad enough when all the dependencies were inside the firewall but XML WS provides wider scope to get things wrong.



Our BSS concept adds the management layer between all applications and the services and components they consume. This allows the infrastructure specific to XWS to be cleanly layered.

The service manager layer makes development assets, be they XML Web Services or components, available to the application developers. Logging, monitoring, QoS, Security and data mapping is all handled in one place allowing UI developers to concentrate on high level business logic.

Does this mean that business rules are handled by the components, the BSS, or both?

Where there is no need for synchronous interaction with a user, the flexible approach is to use the BSS as the

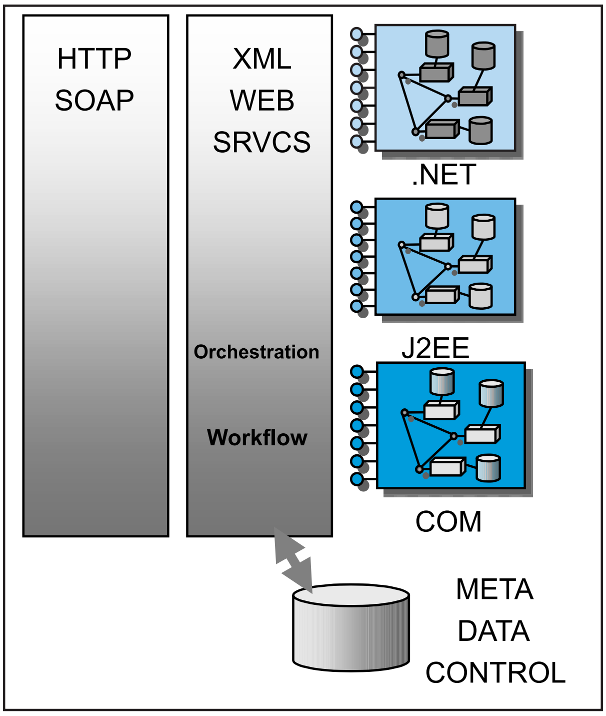


Figure 2: New Layered Architecture

service aggregation and orchestration tool. However, the performance of a direct interaction between the application and the Business Object layer will always be superior and for component assets within the firewall this will always be the choice for speed. ***Tell us what you think!***

Provision vs. Consumption

In our first two reports we started to develop the idea that the BSS needs to provide very similar facilities for both consumers and providers of XML Web Services. The fact that some of the WS stack is still under development (identity, encryption and referral for starters) means the more system code you can leave to the services manager the better.

Is the architecture still 3-tier?

In the scenario below we have in fact blurred the role of transport layer and business logic. The components clearly encapsulate business rules, but the service layer is providing both process orchestration and typically middleware functions such as transport, security and fail-over. Does the services manager become a 4th tier, managing business rules and process or is it simply an adapter providing managed middleware for COM and Java?

The logical conclusion of the 4-tier model is that our business object layer gets demoted to simple CRUD functions and rules are moved up to the next layer; an approach that has been around some time in Business Rule servers.

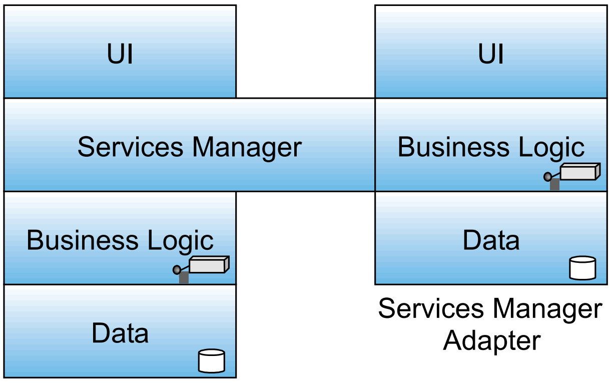


Figure 4: Services Manager as 4th Tier

Architecture for Web Services - Is There Something New Here?

XML Web Services for the extended enterprise presents some exciting new possibilities to get right many of the visions of frictionless e-commerce, B2B and joined up government that have failed to materialize using Client-Server and Web technologies alone. A definitive architecture for its widespread deployment is, in our opinion, subject to some debate and we look forward to your feedback.

Meanwhile, our survey of what software companies are doing to provide off the shelf frameworks continues this month with BEA, Interkeel and AltoWeb.

continues...

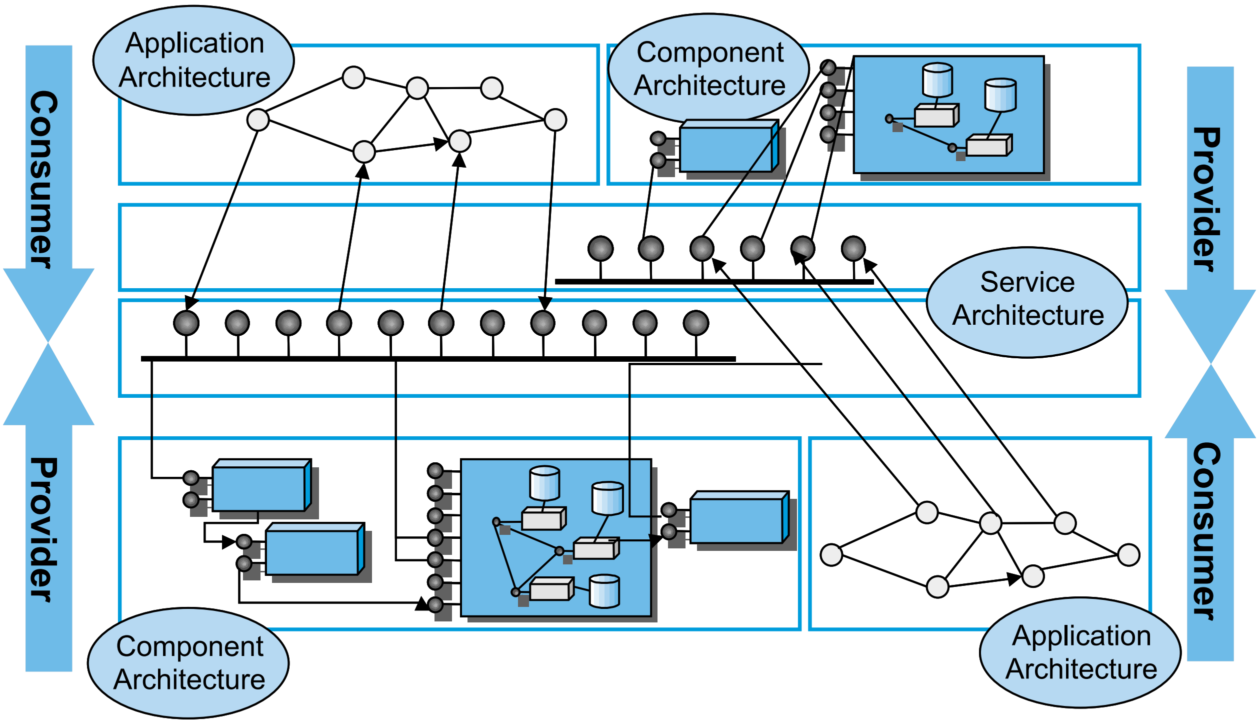


Figure 3: Architectural Reality

Product Update

BEA WebLogic

As a market leader in the application server space, BEA's approach to XML Web Services deserves some careful study. BEA's application server provides the infrastructure that underpins many of the Business Services Server platforms run on; but so far we have not really delved very deeply into what BEA are doing with XML Web Services. We have been waiting for the grand unveiling of Cajun, their new application development tool. This will be paraded at its eWorld Conference in San Diego this month and has been subject to much speculation as to whether it will be capable of building applications for all J2EE compliant platforms or just BEA. So, before we join in with the future-ware discussion, some recent history:

WebLogic version 6.1 was released in July 2001 and this version provided tools to expose Java methods as XML Web Services. The tools can generate Java stubs and provide the necessary SOAP transport layers. Together with WebLogic's support for synchronous and asynchronous message driven beans, developers can SOAP-enable their Java components.

Now for the future: BEA's Mark Prichard spent some time briefing CBDI Forum on the Cajun project and promised us a copy to play with when it is ready so we will provide more detail in time. Cajun is a project that was acquired in BEA's purchase of CrossGain, which was started up partly by some ex-Microsoft engineers. It provides a framework that aims to deliver a VB-like tool that runs on WebLogic and uses the XML Web Services layer to deliver distributed computing benefits. Like Studio.NET it will be driven by a graphical drag-and-drop/property sheet style of development. The main thrust of the first release will be to allow developers to work more at the business-rule level and leave the 'system' code to the framework.

In bringing out this development tool before J2EE 1.4 has been even fully specified BEA will have to invent some mechanisms for marking the 'Web Methods' that may become non-standard once J2EE 1.4 ships. It will be interesting to see how well it conforms to the Axis mechanisms discussed earlier. Clearly BEA has Visual Studio.NET in its sights and we expect so see some significant productivity gains over existing development environments.

Interkeel

Interkeel is a good example of a company who has seen the need for a managed approach to web services. They compete in the same space as two server companies mentioned last month: West Global (mScape) and Primordial (WSBANG). Their expertise is very much based in the Java world and they target BEA and IBM's J2EE servers. It is interesting to note that their broker is being deployed alongside Grand Central and Flamenco at one particular site where the two approaches are seen to be complementary (see http://www.cbdiforum.com/secure/interact/2002-01/implement.php3).

We hope to bring you a full product report in the coming months, but to whet your appetites here is the top-level summary of its features:

- Loose coupling: In the interKeel system, all service invocations are at an abstract interface level. At runtime, the broker then binds the abstract invocation to a service endpoint (or multiple service endpoints). Thus client applications are insulated from changes to service implementations or locations.
- Service mappings and micro flows: Users can normalize disparate services, offering the same function, to a common higher-level, service interface. InterKeel's provides visual mapping and micro flow tools to map the normalized interfaces to underlying implementations.
- Fail-over: Allows failure contingencies to be set-up by defining multiple service providers for the same Web service. If a service times-out, the broker automatically fails-over to alternate providers or endpoints.
- Monitoring and logging: Monitors response time, and load for all services consumed, and enables pro-active detection of performance/availability problems.
- Access Control: By providing fine-grained access control, based on JAAS, the broker enables administrators to centrally control access policies for all services, internal and external, instead of having to administer access at each publishing platform.
- Late binding: Application developers can opt to delay selection of services until runtime, and provide the broker with context and selection criteria for it to do dynamic discovery and binding from services registered in a private registry.

AltoWeb

AltoWeb is an altogether different kind of server product from Interkeel's. It provides a complete J2EE compliant application development framework that runs on top of the leading J2EE servers (WebSphere, WebLogic and Jboss). An integrated suite of J2EE components, Servlets and JSPs provide a rapid application development platform that helps developers build J2EE web applications that conform to the Sun BluePrints for architecture. The development studio accelerates development through graphical tools: debugging, drill-down to component logic and unit testing. By providing ready-built building blocks, development is accelerated and Ford, for example, managed to build a Facilities Management system in less than 2 months.

In addition to the functionality AltoWeb provides ‘out of the box’ you can integrate existing EJBs, databases, legacy applications and of course XML Web services. The runtime monitoring and logging built in to the framework provides some QoS management functions.

We believe that most of the appeal of AltoWeb will be in the rapid delivery of Web (HTML/WML) applications and the framework comes with options to create presentation layers in HTML, XML, WML, Java or SOAP. The approach that sees XML Web Services as another presentation to the outside world perhaps points to their strengths in the Java environment; and resonates with Sun’s view of XML Web Services (i.e. a bolt-on to Java rather than a new platform).

Conclusions

- Invocation of XML Web Services can be done using native HTTP and SOAP but through tools supplied with Apache AXIS and .NET, developers don’t really have go that near to the raw protocols.
- In the Java world the tools are rapidly being enhanced to support point and click style programming using XML Web Services and we will track the new releases as they emerge. Meanwhile, there are plenty of wizards and stand-alone toolkits available to get you started.
- We advocate an approach to the introduction of web services that is encapsulated in our Business Services Server concept. By introducing an extra layer to the standard 3-tier architecture we have added the capability of defining business rules outside of the middle-tier components. We invite your comments on this. Meanwhile, software companies are closing in on this space in an effort to provide a rapid and managed implementation of XML Web Services.

Jonathan Stephenson *jonathan.stephenson@cbdiforum.com*

	RT management				**WS Development**		
	Provider	**Consumer**	**Billing**	**QoS**	**RAD**	**J2EE**	**COM/.NET**
AltoWeb					✓	✓	
Apache AXIS						✓	
Avinon	✓	✓		✓			✓
BEA WebLogic						✓	
*BEA Cajun					✓	✓	
Cape Clear	✓					✓	✓
Epionet					✓		✓
Flamenco	✓		✓	✓			
Grand Central	✓	✓	✓	✓			
IBM WSAD						✓	
IBM WSHT	✓		✓				
IBM WSTK						✓	
Infravio	✓				✓	✓	
Interkeel		✓		✓		✓	
MSFT BizTalk	✓	✓					✓
Primordial	✓	✓	✓	✓			
SunONE						✓	
West Global		✓		✓			

*Future product

NB. This table is a guide to perceived strengths only, please refer to full product reports

Table 1: Products Discussed and Perceived Strengths

IBM Seeks Partners to **Drive Adoption** of XML Web Services

By Lawrence Wilkes

This report looks at two new initiatives announced by IBM this month to help drive customer adoption of XML Web Services. Together with other industry heavyweights, they have formed the Web Services Interoperability (WS-I) Organization to ensure interoperability through adherence to standards. They also announced the Web Services on WebSphere (WoW) partner program to drive the adoption in IBM's direction.

Trying to get a technology they have pioneered adopted as a standard can be a two-edged sword for vendors. They can drive and evangelize the standards initiative only to have customers say, "that's a good idea" and promptly buy a competitor's similarly standards-compliant product instead.

IBM has been in particularly evangelical mode recently with XML Web Services. Having spent nearly two years promoting the concept there is no doubt that the whole industry now has the message, and hopefully in 2002 so will end users.

IBM believes they (and the rest of the industry) have two years to make XML Web Services successful, or it will become just another CORBA type initiative that never quite made it. As such, they feel they must aim for broad customer adoption this year

Web Services Interoperability (WS-I) Organization

To this end, the most recent industry activity is the formation by IBM together with a host of other vendors and user organizations, of the Web Services Interoperability (WS-I) Organization[(1)]. This cross-industry initiative is designed to accelerate adoption of web services by ensuring interoperability across a wide variety of platforms, applications and programming languages.

We see this as a welcome and important move. As we reported in Interact last November[(2)], the number of different web services protocols is beginning to explode. WS-I can play an important role here in two ways. Firstly in co-coordinating multiple standards activities that are taking place across different standards bodies. And secondly, in evangelizing these standards. Our experience is that very few organizations are aware of these activities, and yet they address some of their key concerns about the security and reliability of web services.

By demonstrating a united front, raising the bar on quality, ensuring interoperability, and providing a common mantra for the web services vision, it will help to reassure organizations that now is the time they should start to invest.

Web Services on WebSphere (WoW)

IBM's new challenge therefore is to ensure that as take up occurs, everyone is using IBM products for XML Web Services wherever applicable. This means ensuring that service providers are running their implementations on WebSphere and that other vendor's tools work with WebSphere.

So far, IBM (and Microsoft's) evangelization seems to be paying off. IBM quote a recent GIGA study that shows that 33% of respondents believe WebSphere to be the most important platform for XML Web Services, followed by 22% for Microsoft .NET, with the rest down in single figures.

However, it will take more than current mind share to be successful in marketplace and IBM has initiated the WoW program to help turn their vision and standards leadership position into product leadership.

WoW can be seen as complementary to WS-I. Whereas WS-I is the whole industry coming together to drive overall adoption, WoW is focused on IBM and it's community of partners to do similar for them.

IBM's key objective with WoW is to build a thriving community of ISVs and SIs with tools, products, applications and skills to help customers rapidly and safely build and deploy web services on WebSphere systems. WoW will target partners in the Enterprise Application market, Systems Integrators and complimentary ISVs. To encourage them to join, IBM are providing a number of services such as

- Technical enablement - WoW enabling tools and products, for example helping partners transition to WebSphere 5.0
- Business enablement - Helping partners develop and implement new business models, for example support selling services rather than software
- PR and analyst activity, and marketing to promote WoW partner community and solutions
- Sales enablement, establishment and promotion within IBM sales/marketing channels
- Web services on WebSphere advisory council planning meetings

Interestingly both WS-I and WoW follow a self-certification approach. Though IBM will be providing marketing labels such as “Powered by WebSphere” as a symbol that an XML Web Service is hosted on scalable, reliable platform, compliance with, or use of any technology will only be via self-certification by the partner.

Conclusions

Clearly the bottom line for IBM is to have the WoW community drag through revenue for them. A major legacy of Lou Gerstner was to instill within IBM an understanding that they were much more likely to achieve their objectives with partners than by going it alone, and over the course of his chairmanship there was a major shift towards revenues derived via partners. The WoW initiative will continue to drive them in this direction.

Whilst the WoW initiative is a practical response to customer concerns, it is clearly focused on IBM's concerns and building WebSphere market share. We suggest that IBM will get much greater acceptance of the web services concepts if they also incorporate some of the broader industry issues that are equally important in their customers' thinking. These might include:

1. Ensure a large part of the community is focused on security, trust, and ensuring that service level agreements can be met. These continue to be end-users main concerns.
2. Assist customers to solve the issues surrounding semantic standards. A major preoccupation for customers is the difficulty of establishing common meanings for communication within and between companies. Interoperability requires a focus on vertical industry semantic standards, not just technical compliance.
3. Leverage WoW to get a strong portfolio of WebSphere-based XML Web Services case studies out in the public domain. It is one thing to demonstrate that the industry is behind you, but in our experience customers are looking more strongly for peer referral.

These initiatives are sensible marketing activities that provide a two-pronged approach for IBM. WS-I drives adoption of XML Web Services, whilst WoW drives adoption of WebSphere. IBM is already in a strong position with XML Web Services and WoW should cement this even further.

However, IBM will need to work hard to convince the entire industry that it is playing on a level playing field, and that the entire industry is welcome on an equal footing, including highly competitive systems integrators, hardware platform vendors, and EAI vendors that IBM also competes with extremely aggressively. The WS-I and WoW communities will clearly overlap and IBM must work hard to ensure that any perception of a conflict of interest between the two does not arise.

Lawrence Wilkes *Lawrence.wilkes@cbdiforum.com*

1. http://www.ws-i.org/
2. XML Web Service Standards Update. Interact, November 2001. http://www.cbdiforum.com/secure/interact/2001-11/xml_web.php3

Product Overview - Xara Online Connectable Modules

"The web changes everything", was a comment often made in the dotcom era, and XML Web Services are the next step in the change process. It is not just technology evolution, but the changes in business model that challenge the traditional way of doing things. One example is application development, illustrated here by Xara who provide a pay as you go, on-line development environment for hosted Web Service based solutions.

We looked at Xara Online last year[1] when they made their first 'Web Service Modules' available. These comprised of a set of graphics oriented functions that could be embedded in a web page as remote services. Though they used SOAP to connect to the actual software components at the server end, this was not exposed to the service consumer who uses straightforward HTTP form protocols instead to send the request.

At the time we reported that Xara also had plans to deliver more business-oriented functions in this way, and these have now been made available, called 'Connectable Modules'. These modules enable data-driven functionality to be incorporated into a website without the Webmaster having to install database or any other software on their server as they are hosted by Xara. The whole development process can be run on-line too via a browser.

Connectable Modules

So far, Xara have made four modules available. These are

- Form - Allows a form to be designed a placed on a website.
- Database - Store data collected from the Form Module by connecting it to a Database Module.
- Database Query - Retrieve data from the Database Module.
- Mailer - Email data to one or more email addresses or set up an automated response email. E.g. this could email results of a query

Using these modules is simplicity itself. You create and edit modules, each of which has a set of inputs and outputs. Applications are built by connecting the outputs of one module to the inputs of another as illustrated in Figure 1.

The entire process can be driven on-line from Xara's Module Store (where users manage their collection of modules) or by using add-ons to Microsoft FrontPage or Macromedia Dreamweaver. It is a bit sluggish using the on-line Module Store, though this is a property of the web, not Xara. Using something like FrontPage makes the user interface appear much more responsive.

When a new module is created this is in effect just creating a new instance of an object. Connecting them together creates a new function call that expects just the inputs that were connected. Each instance of the module that appears in the Module Store is effectively a black-box object that performs a given operation defined when the Module is created.

A Connection Layer drives execution of the application. When the process is triggered (typically when someone submits a form, but the trigger can be just about anything) then the Connection Layer process, which knows how and which modules are connected, goes through the process calling each module in turn, taking its data outputs and passing them onto the next module in the process.

Wisely, Xara have also made available a number of sample applications that illustrate how these connectable modules can be used together to support a particular business requirement. For example a

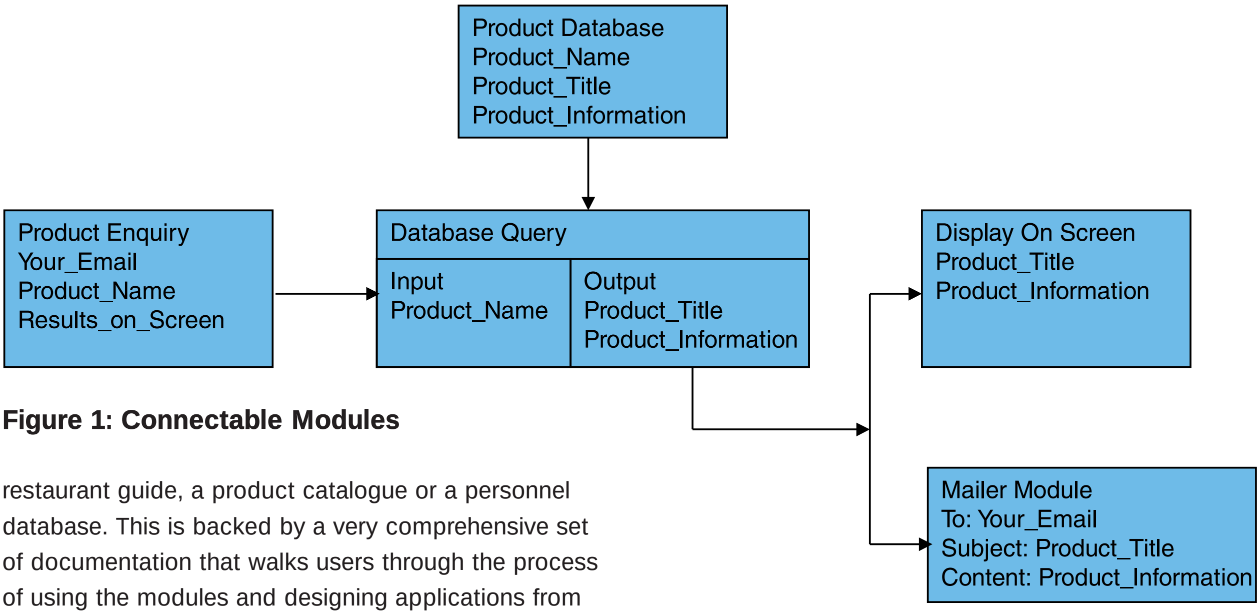


Figure 1: Connectable Modules

restaurant guide, a product catalogue or a personnel database. This is backed by a very comprehensive set of documentation that walks users through the process of using the modules and designing applications from them.

XML Web Services

Where Xara differs from other web application authoring tools, or various form-driven 4GL type tool that it resembles, is the use of XML Web Services. The connections in applications built with the Xara Modules are enabled by SOAP Web Services. When a Module is saved, a new instance of a web service is created, along with it's WSDL file. The Connection Layer even works out what data is required on the inputs and is expected from the reply by examining the WSDL file of the new web service just created.

However, today these XML Web Services not currently exposed to Xara's customers for external consumption, though they plan to do so in future. This is probably not a problem for Xara's current non-technical target market who probably have little need right now for web services, but it is reassuring that they are 'future proofed' for when they do.

Target Markets

Xara believes it's target market is a largely non-technical audience. This might for example include Webmasters who are adept at designing a website, but who have little or no experience in building database applications. Many small businesses will also run their websites on hosts that do not allow them to install their own databases and associated software.

However, I could see many other classes of users finding this a useful way to prototype or build 'disposable' applications. Perhaps the marketing department want to quickly throw up an application to get user feedback on a new product, or you just need a simple internal facing application, but the time and cost of getting the IT department to build this are a barrier.

With Connectable Modules you could have the system on line in no time.

Directions

Xara plan to make a number of other Connectable Modules available in the future. There is for example a Reporting Module and an SMS input module in development (Xara already have an SMS output module too, but this is not yet 'connectable').

One that I quickly saw a need for whilst experimenting, is some way to add logic to the process to enable more complex behavior. Though this would appear to go beyond their target market, in reality the need to make at least simple decisions based on data is a common requirement.

Xara are also looking at Microsoft .NET My Services integration that would seem sensible given it is another set of hosted web services that would complement Xara's Modules well.

Pricing and commercial

As a hosted service, Xara have the opportunity to implement some alternative commercial models compared with traditional software sales. Today there are two basic models provided

1. A free model driven by advertising. This limits the user to quite low numbers of records that can be stored, and also limits some functions.

2. A monthly or yearly subscription model. Provides full functionality and is advertising free.

One expectation of Xara is that independent website designers will become resellers of Connectable Modules as they design website for other people.

continues...

‘Product Overview - Xara Online Connectable Modules’ continued...

Future of Software delivery

So is this the future of software delivery? Certainly it seems to fall in line with some of our predictions; configurable services hosted somewhere on the web, the focus on assembly rather than programming, the pay as you go model, and the automatic delivery SOAP, WSDL, etc.

Right now, the capabilities offered by Xara are not very sophisticated, but there are countless simple applications to which they can be applied.

There are other tools that make the manipulation of XML Web Services easy, and there have long been 4GL type products that make it easy for the non-programmer to build database applications. However, Xara appears unique so far in bringing both these aspects together. Not only can you easily assemble an application from web services without knowledge of the underlying technology, but you can dynamically build the implementation behind those services too, all on-line.

Lawrence Wilkes *lawrence.wilkes@cbdiforum.com*

1. Delivering Web Services to the Masses, http://www.cbdiforum.com/secure/deliver_web_serv.php3

the CBDi Forum

background

The CBDi Forum is recognized as an authoritative source on advanced business practices and IT architectures, and in particular all aspects of web service and component based business. The Forum provides experience interchange and industry analysis for the software industry and its customers. The 10,000 Forum members include IT architects, business analysts and development managers from the world’s major enterprises as well as thought leaders from the software industry. Two principal analysts facilitate the Forum, assisted by several associates, providing a continuous analysis service, special interest group meetings, plus high-level consulting and education workshops.

principals

David Sprott and Lawrence Wilkes are software industry veterans, well-known commentators and industry analysts with long careers in the industry covering many technical and business roles. Both David and Lawrence specialize in advanced application delivery approaches, technologies and markets.

contact the forum

If you have questions, comments, suggestions for future topics please contact us at **info@cbdiforum.com**

CBDi Forum Limited, PO Box 7867, Crowthorne, Berks, RG45 6WD, United Kingdom

subscribe to interact

The INTERACT Journal is published monthly and subscription is included in Silver and Gold CBDi Forum membership. If you do not receive your own copy of INTERACT you may subscribe immediately at:

www.cbdiforum.com

Insight for Web Service & Software Component Practice

IMPORTANT NOTICE: The information available in this publication is given in good faith and is believed to be reliable. CBDi Forum Limited expressly excludes any representation or warranty (express or implied) about the suitability of materials in this publication for your purposes and excludes to the fullest extent possible any liability in contract, tort or howsoever for implementation of, or reliance upon, the information contained in this publication.