MIDDLEWARE**SPECTRA**

incorporating FINANCIAL MIDDLEWARESPECTRA

Contents

November 2002

Volume 16 Report 4

Pervasive computing and business application integration at Arcom and its customers

Arlen Nipper
President
Arcom Control Systems

Management introduction

Arlen Nipper is President of Arcom Control Systems, Inc. based in Kansas City, Kansas and Cambridge, England. Arcom has 100 employees and is a division of Spectris Ltd (where the Group sales are in excess of US$800M) . Mr. Nipper is a computer engineer by training and Arcom's corporate business lies in providing embedded industrial computers — hardware and software — for what is now called 'pervasive computing'.

What differentiates Arcom from its competition is its ability to deliver complete solutions, rather than just the hardware or the software. While Arcom started, in 1982, by supplying embedded and communications processors, it is now much more broadly based: it natively integrates its single board hardware with an operating system (from Embedded DOS, Windows CE or Windows XP Embedded through Linux to QNX, amongst others) and it will write applications as well as tools to build those applications.

In addition, Arcom works with other software products — like IBM's WebSphere MQSeries, WebSphere MQSeries Integrator, WebSphere MQ Everyplace and WebSphere Device Developer (WSDD) Java IDE — to create specific solutions. In this case study, Mr. Nipper reviews how, why and what Arcom has been delivering to customers, using specific examples where appropriate.

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

Setting the scene

The Arcom niche lies in being able to deal with infrastructure for our customers. It is relatively easy to go out and obtain your board, memory and operating system, possibly in thousands of units. But integrating these into an existing application architecture that was created by IT is quite a different issue.

We have no typical customers. Each is different. From our Kansas City location, we specialize in SCADA-oriented (Supervisory Control and Data Acquisition) computing (Figure 1.1). We are systems integrators for oil and gas companies, specifically covering electrical facilities, liquid pipelines, gas pipelines, etc.

In contrast, our Cambridge office concentrates more on controllers for circuit boards for printing machines and for laser printers and other devices. For example, in an industrial environment, Felixstowe Docks in the UK uses our controllers and embedded software in all its container positioning and movement systems. Rather differently, the London Underground's 'Mind the Gap' audio warnings are delivered via Arcom sensors and embedded computers.

Before messaging and integration

Before I start to discuss messaging and integration, I would like to look at the practical environment in which we often find ourselves operating. In particular, it is my experience that IT and embedded systems people — although employed by the same organization — do not necessarily work together. To understand why, and the implications, let me describe how our systems work.

Over the last decade Arcom has been known for its industrial network gateways. We took the boards that we built and deployed these into specific customer settings. Indeed, one of the distinguishing services we offered was the capability to understand a particular customer's existing (legacy) SCADA topology and then be able to work with this to update it — using our single board computers and our industrial network gateway.

I would say that today, although there are no firm statistics on this from independent oil companies, 30%-40% of the advanced liquid pipelines in the US use our products to provide their SCADA infrastructure capabilities — the receipt and transmission of data:

- **from remote (embedded) devices placed alongside pumps and the like**
- **to some form of control center**
- **and vice versa.**

In essence what we have been doing is taking any existing SCADA information and, via our industrial network gateways, enabling that data to be communicated over newer network topologies, like frame relay and/or TCP/IP (Figures 1.2 and 1.3). We have done this in conjunction with oil, gas and electric industries and used VSAT network infrastructure provided by AT&T, GE SpaceNet, Hughes, etc.

Our success with this was built upon an architecture that we created some 10 years ago. This was designed, from the outset, to be able to handle the many native protocols that you find out in the field. The need was, and is, to interface to all types of sensor out in the field — irrespective of whether these are flow computers, gas chromatographs, tank level meters, RTUs or even PLCs.

Traditionally, this information moved over dedicated networks into a customer's operational computing center to be processed by a single dedicated application that was purpose built to consume the data gathered from the field. In practice, each organization had its own proprietary network, carried over VSAT or X.25 or some other communication mechanism. Often the consuming application ran on a dedicated box with no integration to other parts of the enterprise. This was not just a dedicated system: it was a dedicated infrastructure.

So, in an oil company-type environment, there might be a pipeline running from A to B with various pumping stations. At each of these stations you would have controllers — for example, metering throughput. That information would be fed back over a network to an industrial network controller which would perform some form of data translation so that the designated application (which was almost always hand built for the purpose) could consume the data.

Furthermore, this was not necessarily uni-directional — carrying data from the field to the control point. That same network could have that dedicated application transmit instructions out to the field equipment — to close a circuit breaker, open a valve or change flow rates through a pipeline. Broadly the same applied situation applied irrespective of whether this was in an electricity utility or some form of manufacturing.

Poll spoofing and rudimentary publish and subscribe

What we found was that, time and time again, the information that our embedded controllers were gathering for our customers belonged to the operational side of a com-

pany, not to IT. Furthermore, the operations and IT people did not usually understand each other. Indeed, the embedded controller and SCADA dimension was foreign, not least because it was almost always uniquely built as a one-off for a particular situation.

For example, an oil company's business is obtaining oil, conveying it to refineries, industrially refining it and then distributing the finished product out to the commercial market, to gas stations, as heating oil or whatever. What we found consistently is that there was a split between:

- **what an organization does for its business**
- **how the information associated with this is managed.**

We felt this was senseless. To try to overcome this separation we built a solution on UNIX — a host communications processor (HCP, in Figure 1.4) — which would talk to all the devices in pumping stations or wherever. This host communications processor would accumulate information 24 hours a day, 7 days a week in an application that Arcom wrote.

Now remember that we were, then, primarily a hardware manufacturer. But even then we had to be able to write some back end software to connect to other systems. The way we achieved this was by what we called 'poll spoofing'. The attraction was that this enabled legacy systems to think they were still polling over traditional phone lines when in fact they were polling one or two feet away to our host communication processor. By this mechanism we were able to deliver data to applications.

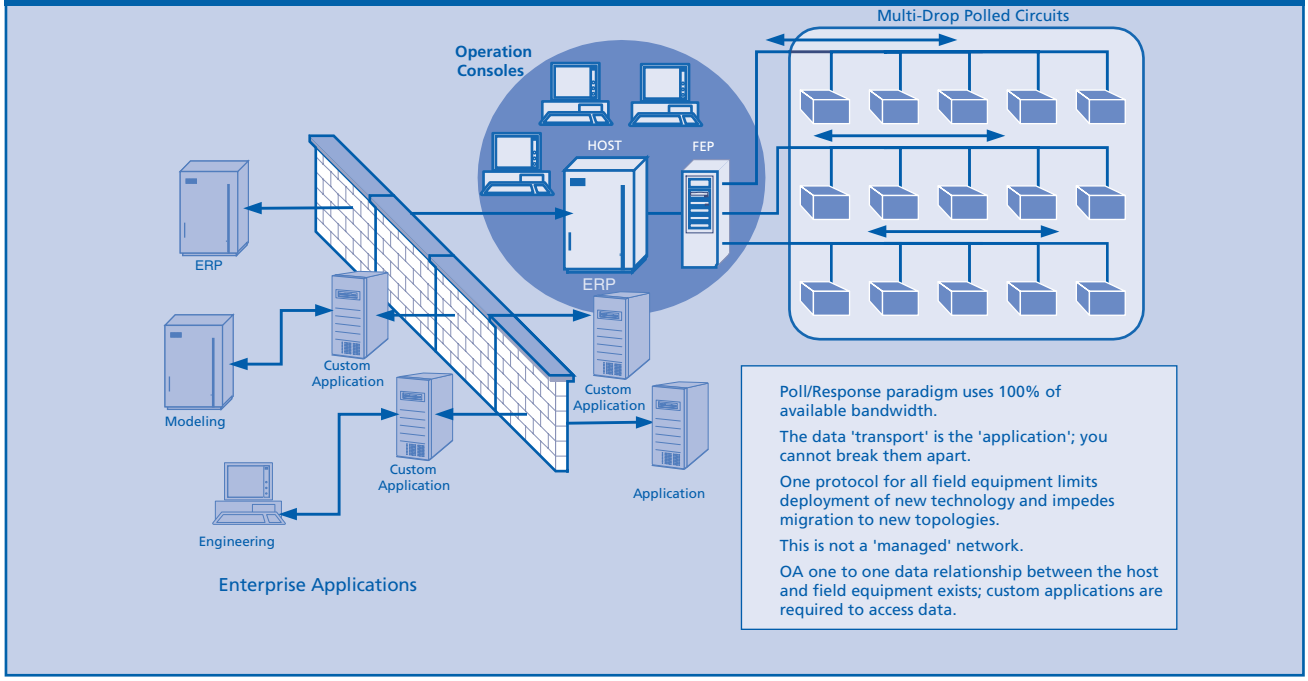
In effect it was a rudimentary form of publish and subscribe (although we called it 'reporting by exception'). The only difference was that — unlike publish and subscribe as we know it today, which is a one-to-many offering — this was pretty much an implied subscription where we knew that we would always send it back to this one box.

This worked well. But it was only the first of several steps that we have taken as we have moved towards integrating business applications and operations.

Making progress at Phillips Pipe Line Company

In practice, Arcom has always been constrained in its business model by one simple factor. We could only undertake those projects where we knew we would be allowed to become intimately involved with our customer's organization — so that we might understand IT plus operations. Only if we could do this did we feel we could deliver the interfaces to the existing devices that were necessary, and then provide the generated data to the data center.

Figure 1.1: Legacy SCADA/Telemetry system model



Yet, we also saw from these projects that much was being repeated within organizations. This, to all intents and purposes, was unnecessary duplication.

About six years ago, Phillips Pipeline Company (an existing customer for our embedded devices) came to us. It had just installed the R/3 application from SAP. R/3 needed lots of data. But there was no obvious way to obtain this data from the existing (legacy) SCADA infrastructure and to automate its entry into R/3.

Phillips Pipeline Company (PPC) came to us and said: "your embedded processors are the only place where one can see the 'data flying across the wire'. Right now you are delivering this data to one fixed application. What we would like is to insert 'data crackers' on the wire so that — as the data flies by — we can 'crack it off the wire' into a form that another application might use to feed R/3."

At that time we preferred, where possible, to try to limit the amount of custom application development that Arcom undertook (even though we had developed several specialist tools to accelerate delivery). One reason for our reticence was the maintenance implication. Every time a customer changed a piece of field equipment — or a newer device came along — we found our programmers being hired back to write additional code or amend existing code.

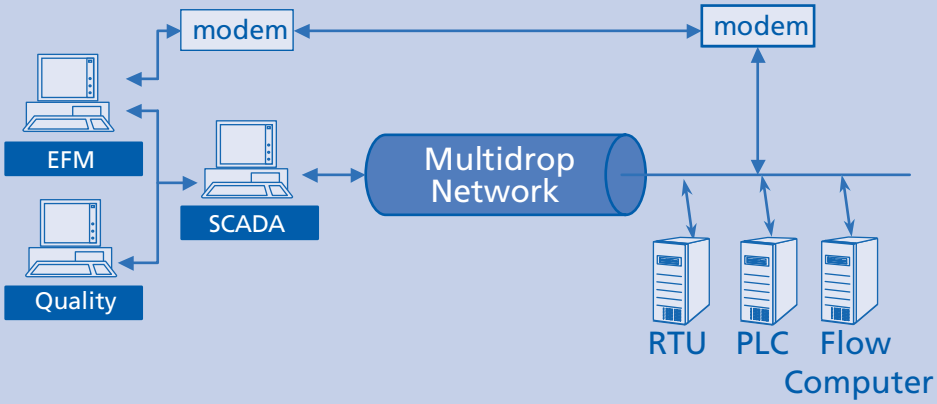
My first reaction to PPC was that this would be much the same. I said 'No', we really are not interested. Its response was bring in one of its IT architects.

Now this was unusual (remember this is six years ago). Operational SCADA departments did not bring in an IT guy. But PPC had already encountered the problem of delivering data between different application silos. It had investigated and it had a solution in mind, from a California-based VC-funded company called Open Horizon — which had produced a Java-based, lightweight, publish and subscribe messaging system called Ambrosia.

Initially Open Horizon thought that it did not need Arcom. Its solution was to put Sun workstations out in the field and run its Java publish and subscribe capabilities to send and receive messages to a control center. When we pointed out that nailing Sun workstations to telephone polls in Texas, or oil pumps on the North Slope in Alaska, was a mite impractical, its people soon understood.

Once the parameters were clear, we had a serious talk. I was, however, somewhat sceptical for we had tried to use a publish and subscribe solution before, using TIBCO's TIB. Unfortunately, the TIB protocol was too heavy for VSAT messaging (where an extra byte per message can cost $1M/year). In fairness to the TIB, it was designed for LANs and fat pipe networks where communication overheads

Figure 1.2: Classic multidrop network



do not matter in the way that they do in VSAT or 300 baud networks.

When the Open Horizon folks started showing us the Ambrosia protocol, we recognized an excellent execution vehicle for our purposes. In fact, if I remember rightly, the Ambrosia protocol only had 18-20 bytes of overhead in the actual control of the messaging.

Working with PPC and Open Horizon, we implemented the client side on our controllers. These sent messages to, and received messages from, the Ambrosia server which then communicated with the mainstream IT systems. What was noticeably different to the TIB model, where every device has to be both server and client, was that in Ambrosia there was a clear separation between the lightweight client and the fully fledged server.

To make a long story short, we added publishing and subscribe to our TCP/IP stack on the industrial network gateway that we had deployed at PPC. A pilot test was installed at the Kansas City fuels terminal:

- **a batch of jet fuel was delivered to the depot**
- **the flow computer registered this delivery**
- **it informed the embedded controller, via an electronic ticket**
- **this ticket was published by the Ambrosia client to the Ambrosia broker (server)**
- **the Ambrosia subscribed to the ticket information and then, in turn, published it to the IT network.**

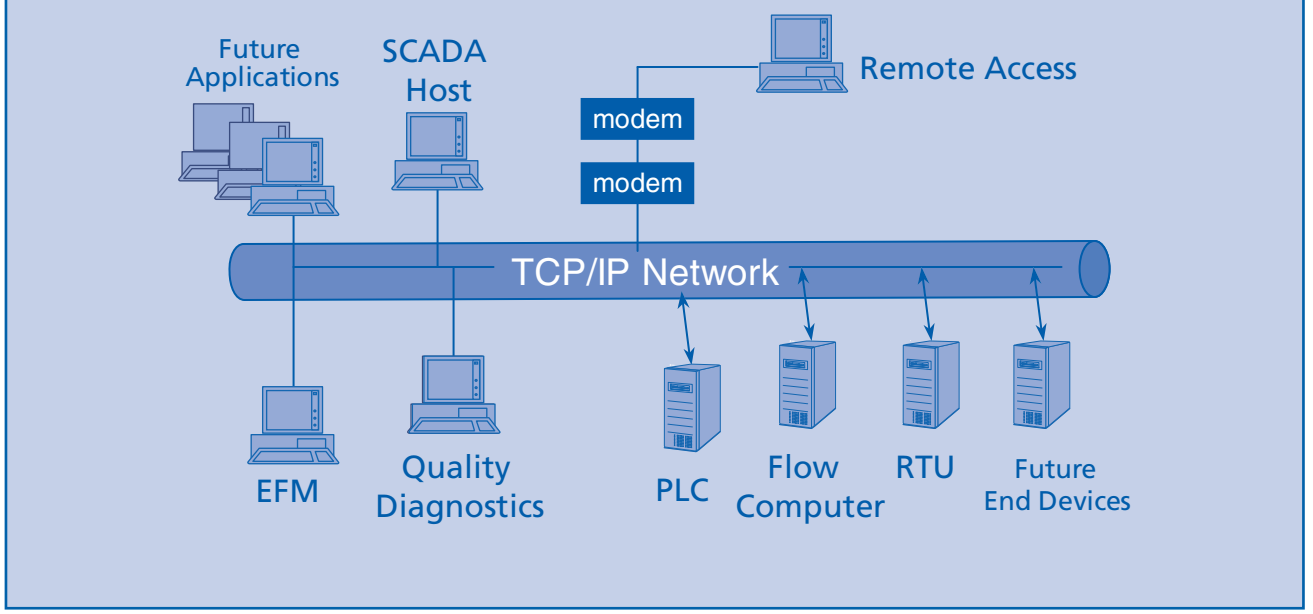
Low and behold, a spreadsheet that our customer had on his desktop subscribed to the data — and the fuel arrival data appeared on the desktop. It had all worked.

Now one thing I have not mentioned previously was that about this time (six years ago) the satellite companies had started putting TCP/IP on their networks. This mattered because we now possessed a common, standard transport protocol, rather than the myriad of in-house, custom built ones.

Yet, at the same time, there was still a huge number of locations which performed connections via dial up. Not only did we prove that lightweight messaging would work via satellite but we went on to prove that it would work over 300bps dial up. We now had frugal messaging operating in a publish and subscribe mode with low byte overheads.

This was neat. We deployed the solution for PPC for its Automatic Meter Reading. Our embedded controllers, running the Ambrosia client, would publish the flow tickets.

Figure 1.3: Enter TCP/IP



The Ambrosia server would subscribe and then itself publish the results out to the appropriate IT applications. The result was that Phillips now had a kind of split data stream:

- **the existing operational data was still being fed to the original legacy application**
- **in parallel, the new TCP/IP publish and subscribe mechanism enabled us to send essentially the same information (if 'wrapped' differently) into the enterprise IT part of the PPC organization — because the subscriptions could pick up the data that was being published.**

News of this got out. I exposed it to several other pipeline and electricity companies. Everybody looked excited about the technology. It was agreed that there should be some form of standard way of doing this within industry.

Then, just when we were ready to take it to a standards committee in Dallas, Open Horizon folded. We had a good solution, lots of interest but no access to the underlying technology. We were stuck, and PPC was left with what became (in effect) yet another legacy solution.

Enter WMQI and IBM

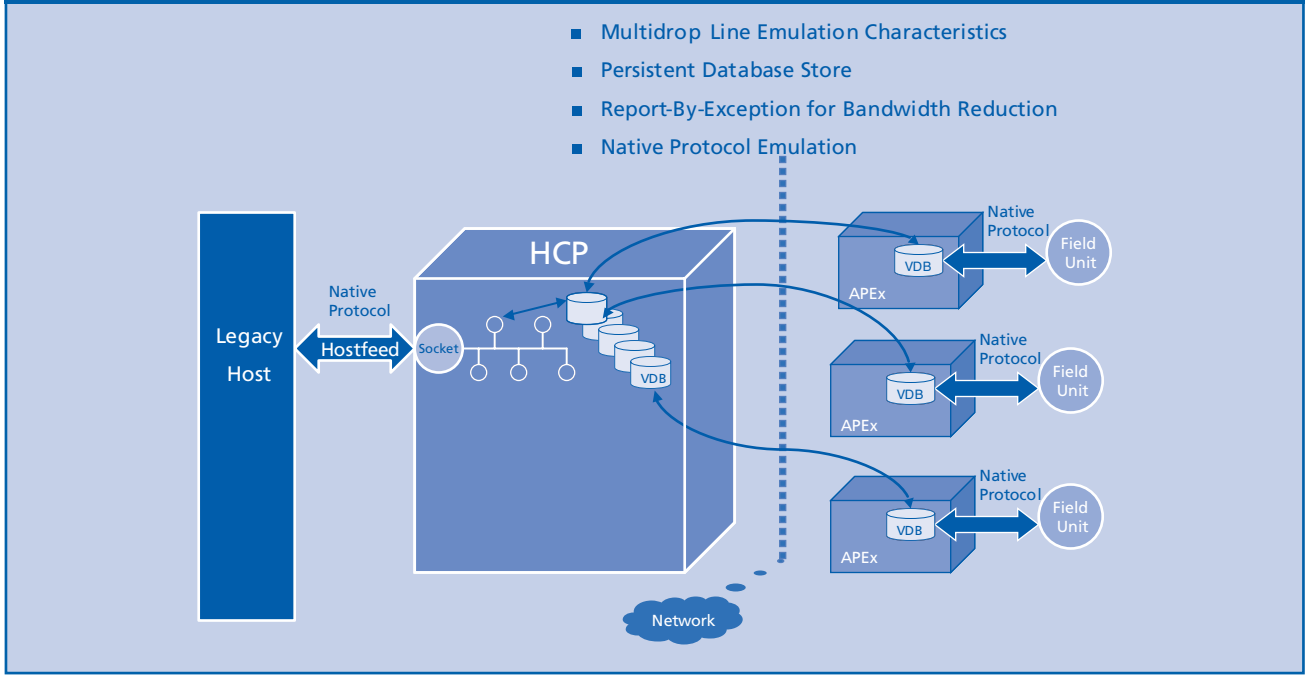
About six months later, IBM called in the person of Tim Holloway (an IBM Distinguished Engineer). He had heard about what we had done at PPC and invited us to the Hursley Lab. in the UK, the home of MQSeries (now WebSphere MQ or WMQ) and MQSeries Integrator (now WebSphere MQSeries Integrator or WMQI).

We visited and learned about IBM's pervasive computing strategy — which meant nothing until we understood it was about embedded computers, merely renamed. The IBM approach at that time was to use WMQ and WMQI (its messaging broker) to tie lots of IT-style (rather than operational SCADA-style) applications together. IBM's new interest was focused on delivering connectivity between IT and operational SCADA devices.

This was a revelation to us. At last a major vendor was going to step up to the table and accept that corporate computing did need a mature infrastructure by which to connect embedded computing. Up until this point, every customer that I had dealt with had used almost a hobby shop approach to embedded computing. Every solution was a one-off, even within one organization. The same function could be developed multiple ways in different places by different people. By the time Arcom was ever called in, there were often multiple parallel solutions.

Take for example one of the major gas distribution companies in the UK. It currently utilizes an Arcom industrial

Figure 1.4: Host Communications Processor



gateway controlled infrastructure to control the delivery of gas. When we first arrived, the legacy infrastructure comprised 12 geographical regions using 18 different protocols going to 14 different back end systems. Now, everything is delivered to a single back end system via a common infrastructure.

The key point here is the one that IBM had appreciated, that industry has to quit treating solutions that include pervasive devices as unique point-in-time solutions. We all need a coherent infrastructure approach to link pervasive computing to mainstream IT.

Nodes and WMQI

What IBM proposed was that there be lightweight input nodes to its WMQI broker. These nodes would receive data from embedded controllers and enter it into WMQI (Figure 1.5). There the data could be transformed, routed and processed to as many destinations in the traditional IT world as were required — thereby breaking the one-off development cycle. Furthermore, adding or maintaining IT function was decoupled from what needed to occur in the operational SCADA network.

With this as the working basis, we went out to develop an WMQI node. When complete, we called this the WMQI STP (Straight Through Processing) node.

In practice this leveraged what we had learned with Ambrosia, except that we made the solution even lighter, and exploited what was not in the Ambrosia protocol. For example, the protocol overhead today is 2 bytes, once you have the session established. But it had been 30-40 bytes.

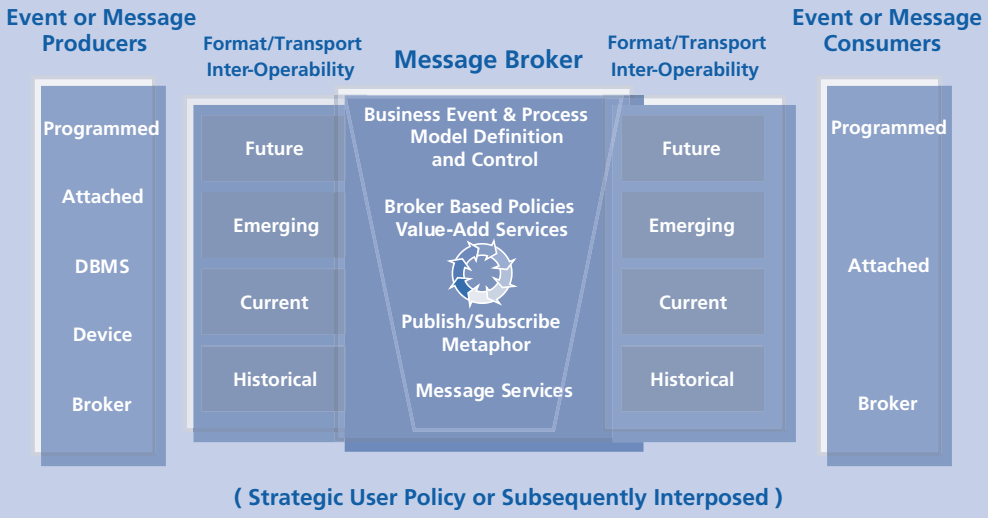
The reduced number was delivered only after I had explained the implications of a project on which we had worked with GlobalStar, where we used a satellite system and 10,000 flow computers delivering a report every 15 minutes. With 30-40 bytes of overhead for each communication, the satellite network charges would have been massive, because the satellite networks charge by the byte. Given these numbers, over a year, each extra byte could cost millions. Suffice it to say, we went back and rethought, which vastly improved the RoI.

What is so good about this is that we can deliver a standard approach to any industry using embedded computers that can receive or send messages which WMQI can:

- **pick up or send out to embedded devices**
- **process (including transforming or routing) to multiple mainstream IT applications systems via WMQ.**

Put another way, the infrastructure approach with WMQI provides:

Figure 1.5: Message broker



- **connectivity**
- **protocols, including publish and subscribe**
- **transformation**
- **routing**
- **standardization of the operations/SCADA to IT interface**
- **scalability.**

Now a customer can obtain what he or she needs (Figure 1.7). We can provide a solution which spans fast networks, medium networks and even slow ones — all through a common point integration point. Another reason we like MQSI STP is that it will run just as well over:

- **a 300 baud network**
- **as over a Gigabit fiber network.**

A Large Consumer Testing Organization (LCTO)

Now let me use another illustration of what can be achieved, using a different (if anonymous) customer that I will call LCTO. This is a large consumer testing company in the States. On average it runs anywhere from 500,000 to 800,000 tests per night. These start from samples sent in by customers and the testing runs from six at night until three in the morning. Delivery of results has to occur between 8-10am each morning.

In essence LCTO has three tasks:

- **the first is to receive samples**
- **the second is to test them**
- **the third is to make sure the customers receive the correct results of those tests.**

LCTO approached IBM and said it had two business challenges:

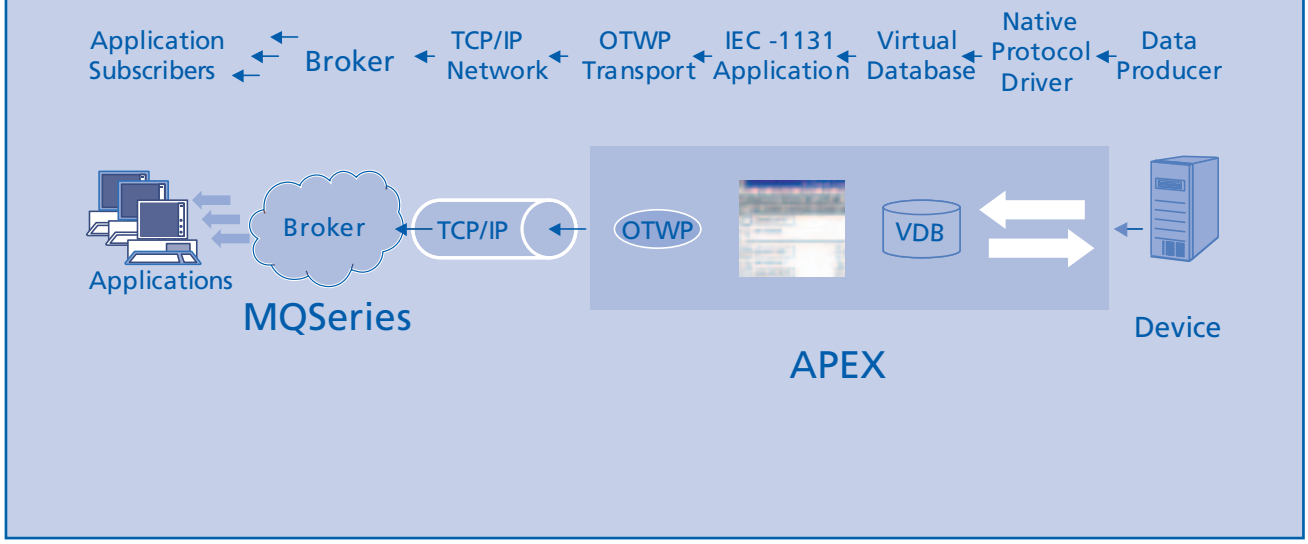
- **the cost of the phone calls to deliver the electronic test results to the customers**
- **the need to be sure that the results were delivered.**

If the first was a cost issue, with millions a month being paid to telephone companies for calls from the LCTO's computing center to all the customers, the second was about quality of service. When the LCTO's computer called a customer's office, it did not know if a printer had run out of paper or ink or simply malfunctioned. As a result it did not know if its reports had been reliably delivered.

It sought to change all this. It wanted a very different approach by which:

- **to put a smart device by the printer in the customers' offices**

Figure 1.6: WebSphere MQSeries Integrator



- **that smart device would be able to call into a local POP number, so LCTO did not pay for all the telephone calls**
- **to implement a quality of service capability such that it (LCTO) could know whether the report had been delivered to the customer's office and been printed.**

To give some sense of scale, we are talking about 50,000 customers' offices which had printers that LCTO wished to replace. IBM approached us to assist in designing the appropriate boxes.

The way we approached the project was first to decide that we did not want to use FTP to communicate the reports (not least because one customer's office might have several reports covering several consumer test situations). Instead, we wanted to deploy some sort of messaging system — a kind of 'all-electronic' FedEx, with your report package being electronically signed for. In addition, we needed to implement a system that would track the report from when it left the lab testing system all the way through until it arrived on the customer's printer. Not only that, we felt that LCTO needed to be able to report back at what time the report printed or, if it did not print, why not.

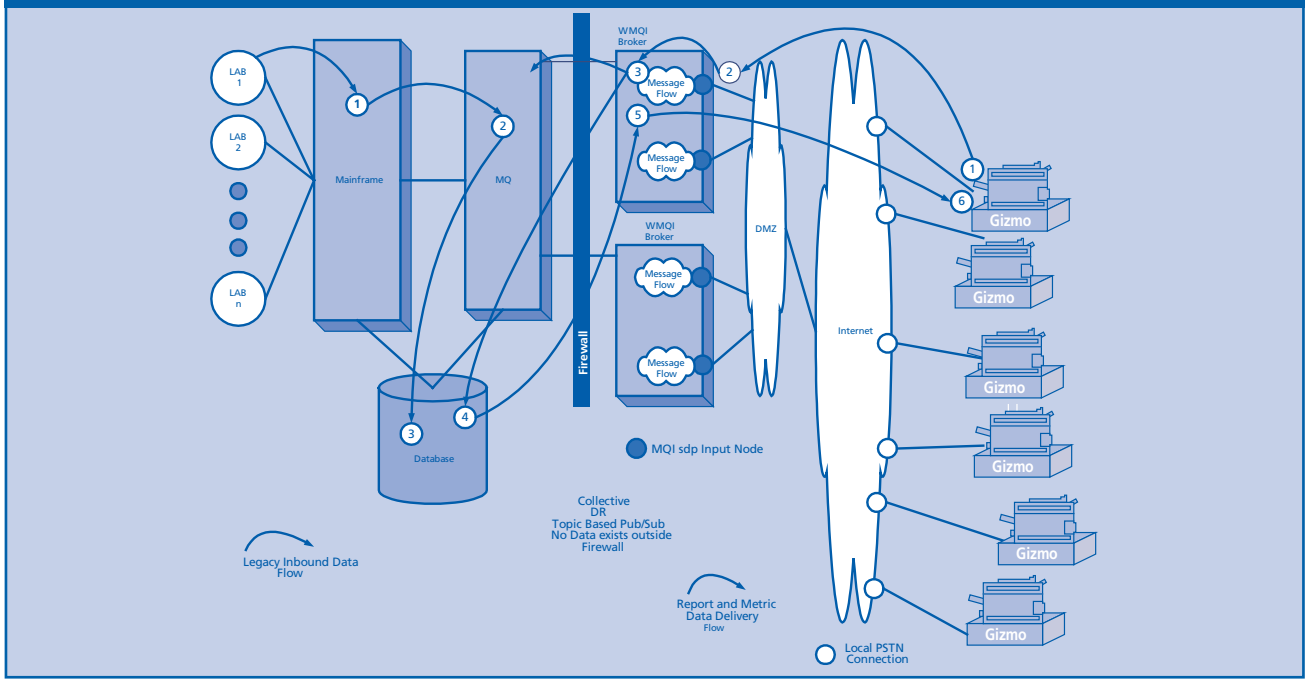
To do this we designed a custom embedded computer, called the 'Gizmo'. This is a small x86 compatible machine with 8MB of flash memory and 16MB of DRAM, a parallel port, a serial port, a USB port and an embedded operating system. It is just a flat, one inch tall slab of plastic that you can set up underneath your printer or put wherever you want.

Then we started to look at the business side. LCTO had a mainframe at its corporate facility. The results of the tests are delivered through the night to the mainframe which processes them and checks that the tests undertaken were the ones each customer had ordered. Finally, the data is processed into a form that can be delivered as intelligible results in the two hour window between 8.00am and 10.00am.

When we first talked about the design we assumed that queues were applicable. In my mind IBM's WMQe (the lightweight MQSeries Everyplace sibling of WMQ) was the obvious solution. So, for about three months, we proceeded using WMQe as the entry/exit point to and from WMQI on the mainframe. WMQI would send out the reports to be received by MQe running on the Gizmo. These could then be printed.

Then a problem arose. As WMQe worked on a queue by queue basis, it was going to take a huge amount of WMQI processing on the mainframe at LCTO to send out all the reports. In addition, the administration of so many queues and static IP addresses — at each end — was going to be massive. On top of this, there were issues concerning roll

Figure 1.7: LCTO's solutions



out: how were we and LCTO to be able to deploy and set up in 50,000 customer offices, with many different sorts of printers, etc. Finally, it was difficult to envision a practical disaster recovery scenario that was workable and affordable. Basically, at the end of the day, using MQe with WMQSI required too much central processor power to manage the queues and the subscriptions — even before deployment. We decided that, in the fixed location scenario that LCTO's customers' offices represented, it was the host environment that was not financially viable.

Having realized this, we had to think about redesign. Our preferred approach now uses WMQI sdp (SCADA device protocol). The difference was a reduction in the CPU processing power of about 80-90%. The first reason is that we do not have to have WMQe gateways. Then there is the fact that an WMQSI sdp node can maintain many more simultaneous connections than an WMQe node: an MQe node maxes out at supporting about 500 connections whereas an WMQI sdp node quite happily will support 2000+ connections. Finally, using datagrams directly over IP for the messaging incurs much less overhead than the more fully functioned WMQe.

The key difference here is about mobility. WMQe was designed for mobile applications, for supporting PDAs and laptops on the move — with all the special characteristics demanded by mobility. If we had been sending data to customers moving around a facility or city, at variable times, WMQe would probably have been the only secure, reliable communication mechanism with the appropriate quality of service. However, LCTO's customers' offices are static.

From a customer's office viewpoint, the Gizmo:

- **dials up the LCTO mainframe**
- **establishes its session with the WMQI broker**
- **publishes its request to any WMQI broker that is listening (and there may be several online in order to support the volumes); the WMQI broker that responds then goes to fetch the reports from the mainframe database**
- **subscribes to the appropriate results (published by the WMQI broker)**
- **prints the results in the customer's office**
- **terminates the local call.**

To achieve this, there is an interesting mechanism that we designed into WMQI sdp. We call this a 'last will and testament', which is lodged when we establish sessions. This says that, if I should die, I want you to deliver this data on this topic for me. We have really used this to ensure delivery to LCTO's customers — thereby delivering the quality of service intended.

Lessons learned

My first lesson is, do not expect operational computing (like SCADA or other forms of process control) to be in close communication with mainstream IT computing. In my experience this is relatively rare, not least because so much process control and its associated embedded computing does appear to be one-off. This need not be the case and I would strongly recommend working to bring the two parties together — as Phillips Pipeline Company did.

The second relates to the first. Embedded, or pervasive, computing should be an extension of the overall technology infrastructure. It should not be sidelined or separated.

Thirdly, by using tools like WMQ and WMQI with specialist nodes like the sdp, it is quite feasible to integrate bi-directional communication. This can be hugely valuable with an immediate RoI. Just think of how much LCTO will save by displacing its long distance phone calls as well as removing the costs of uncertainty (about whether a customer has received the test results ordered). Similarly, saving $1M per byte per year is feasible in certain satellite network configurations — by selecting the appropriate technology.

My last lesson is that we — Arcom, IBM and others — are still educating the industrial market. Phillips Pipeline Company, Chevron, Automated Energy and others are visionaries compared to much of industry. The power of embedded processing, whether in mobile PDAs or tablets or in fixed location controllers, is only just being appreciated. IT needs to learn more about 'difficult' communications environments. Operations needs to learn the skills of IT which avoid re-inventing the wheel for each new project. The scope for integration is immense, if organizations seize the opportunity.

Management conclusion

The experiences of Arcom Controls, derived from multiple customers, show how integration can be delivered, via a range of middleware, both inside and outside conventional IT boundaries. At the same time Mr. Nipper emphasizes that, if this is to happen, a broad, rational approach is needed. The past divisions between SCADA-type infrastructure and IT infrastructure were clearly not optimal. Their result was parallel infrastructures. This is no longer sustainable, in any industry. With intelligent use of middleware, practical solutions are at hand.

Exploiting work flow and middleware at Centerprise

Eric Yu and Dean Keister
Chief Technology Officer and
Director of Networking Architecture
Centerprise Services

Management introduction

Centerprise Services, based in Purchase, NY, was founded in late 1998 by a number of Wall Street veterans, many of whom came from the Union Bank of Switzerland (UBS) after its merger with Swiss Bank Corp (SBC). One common thread among the founders is that most had worked in the trading, risk management, risk control and product control functions, including risk analysis, risk reporting and regulatory requirements.

Today, Centerprise has built a suite of applications called CenterSphere(tm), which is aimed at the financial sector, particularly wholesale finance organizations. The architecture of the solution, oriented towards service provision in an ASP business model, has evolved into something richer and more powerful, which as it turns out provides the support needed for enterprise-scale application deployments, whether via standard internal deployment or ASP approaches.

Eric Yu is the chief technology officer of Centerprise. Dean Keister is the director of networking architecture. They discuss the design, architecture and middleware, including the WebSphere family, that they have used to create both functional applications as well as work and process flow dimension to guide decision-making.

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

Qualitative > quantitative

When we left UBS, we saw a space in the market where there was a need for not only quantitative risk analysis tools but a complete platform that would address risk on a holistic, enterprise basis, especially that experienced by capital markets firms. In effect, we saw the requirement for an integrated application suite that would be capable of providing the full range of management and control functions that today's capital markets firms need. Another way of putting this is that we aim to be the equivalent of ERP for financial services. That is not to say that we plan to go head-to-head with SAP on financial reporting or general ledger. Rather, we intend to provide an integrated suite designed for the functionality specific to the wholesale financial services industry.

At the same time, we intend to deliver an applications environment that embraces what the operational management and risk control needs are within these organizations. This means focusing on process management. We believe a key piece of our value proposition is not just the quantitative risk control and risk measurement pieces, although these are important and we do provide them. From our industry experience, we realized that the qualitative control aspect of risk tends to be missing from most risk management solutions and is also missing from the general operational environment in most financial firms.

Qualitative control is really important. Indeed, we stand by the adage that 'good risk management is the same as excellent process management'. In particular, regulators are invariably keen to see that due process has been executed consistently. One has only to think about the changes that the Basel II Accord will bring to understand.

As we started to think seriously about how to address qualitative risk control, which includes quantitative risk control, we realized this is an issue running through most enterprises. As a result, work flow management and process management have become the cornerstone of our overall enterprise platform. Qualitative risk reporting, via process management, now extends all the way to supporting corporate accountability.

We also realized the importance of modularity. We are now concentrating on providing an integrated application suite, which is modular, so that our customers can buy what they need for their particular financial firm when they need it. On top of any of our modules, you can always include our work flow and process management functionality to address, and avoid, the type of control failures and costs that financial firms typically face. Through our CenterSphere Suite, our aim is to improve quality and reduce costs for customers, including enabling them to use existing applications where appropriate.

CenterSphere

To appreciate what the CenterSphere Suite offers, you need to understand the role and place of middleware. Middleware technologies provide the infrastructure upon which we have built the Suite (see Figure 2.1). In this context, we should describe the various modules, which are:

- **OrganizationCenter: this covers organizational management and key human resource data**
- **PortfolioCenter: this provides portfolio management**
- **CenterFlow: the work flow element**
- **MarketRiskCenter: this provides market risk management and control**
- **CreditRiskCenter, which functions the same as MarketRiskCenter, but for credit risk**
- **OpRiskCenter: the same as MarketRiskCenter, but for operational risk**
- **CenterCalc: the model control piece that manages instrument-level analytics**
- **CenterFinancials: essentially, the General Ledger**
- **RelationshipCenter: where customer relationship and counterparty reference data are managed**
- **TradingControlCenter: this provides the daily revenue analysis and product control**
- **LegalCenter: this offers compliance, legal and regulatory capabilities and policy management**
- **AuditCenter: this provides audit management**
- **ConnectionCenter, where data collection and monitoring occurs**

They are tightly integrated, yet modular, as we will describe later; all are designed to be operationally efficient and resilient. Being modular, institutions can purchase only those components they need to meet their operational and RoI strategies.

Architecting the CenterSphere Suite

So, having described where we are coming from, let uss move on to how we are delivering. Our platform has been designed around:

- **Java**
- **JavaBeans**
- **J2EE**
- **a Java-based application server.**

Why did we go down this route? A key goal was that our CenterSphere application suite be able to run over the Internet or over an intranet as a series of services that could be made available within a wholesale financial institution as well as out to third-party institutions like clients, customers and partners. As such, Java, the use of browsers and an application server were pretty much a choice already taken for us. Indeed, we also wanted to be able to deploy via the ASP business model that was in vogue some three to four years ago.

Just having to consider how we would deliver high availability and reliability in a Java-based environment operating over the Internet obliged us to think through what was needed. In particular, the ASP approach made us work through the modularity issues in depth, plus all those other issues associated with being able to distribute function across two or more systems.

Today, however, large wholesale financial firms still prefer the standard software licensing model. This is now our main delivery option, although the inherent modularity and Java capabilities means that we can always return to the ASP approach if there is demand for it. Under either delivery model, financial institutions want their customers to be able to access their services and products over the Internet. What we provide are the applications, linked by an infrastructure built on established middleware, to satisfy both end customers and financial institutions.

Indeed, we see a third dimension. Increasingly, we find large financial institutions are looking at their existing internal IT infrastructure. They would like it to resemble a 'mini-Internet' when it comes to delivery of products and service capabilities. They want cross-business, cross-product, cross-boundary, cross-application and cross-function capabilities enabling them to deliver what their customers desire.

However, most of the applications that these firms are running have been designed and built in isolation from each other. This makes integration, never mind the control of both quantitative and qualitative risk, much harder. The appeal of our architecture is that all our modules:

- **have been designed to work with each other**
- **can also work with existing applications, where necessary, although significant work is needed to match an existing application with the capabilities found in one of our already integrated modules.**

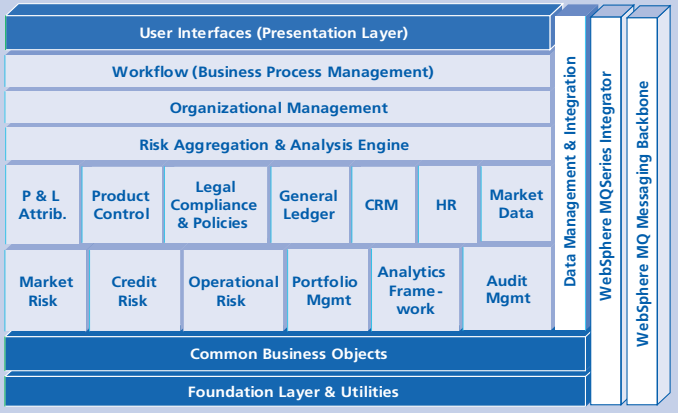
The result is not the monolithic, unmaintainable monstrosity that everyone fears. As it turned out, attempting to satisfy an ASP model from a technical perspective compelled us to introduce the separation of application functionality from the infrastructure. This has an interesting follow on. Once you commit to implementing an inter-business solution, as a technical ASP model invariably enforces, you have to face many implications that go well beyond an intra-business solution. Put another way, if you can do inter-business, you can very likely do intra-business, within one organization or firm. The converse is not necessarily true. Being able to deliver an intra-business solution behind an organisation's 'own walls' is often not susceptible to being extended through 'those walls'.

The middleware and infrastructure dimension

Once we had shaped what we thought our solution should look like, we confirmed our decision to use a Java-based approach. From there, we needed to choose a platform, in this case an application server. We did our due diligence and looked at a number of vendors: BEA/WebLogic, Gemstone, IBM, etc.

Looking back, there was really only one choice for us, given that our center of focus is the wholesale finance sector where longevity and commitment matter. This was IBM's WebSphere, even though it was then, a couple of years back, not in nearly as good a technological shape as it is today. Nevertheless, although it was probably a half generation behind at the time, we were comfortable that:

Figure 2.1: Middleware in the CenterSphere Suite



- **the WebSphere Application Server (WAS) would catch up**
- **financial customers would be similarly comfortable (after all, most of them are already substantial IBM users)**
- **we would have the support we would need from IBM.**

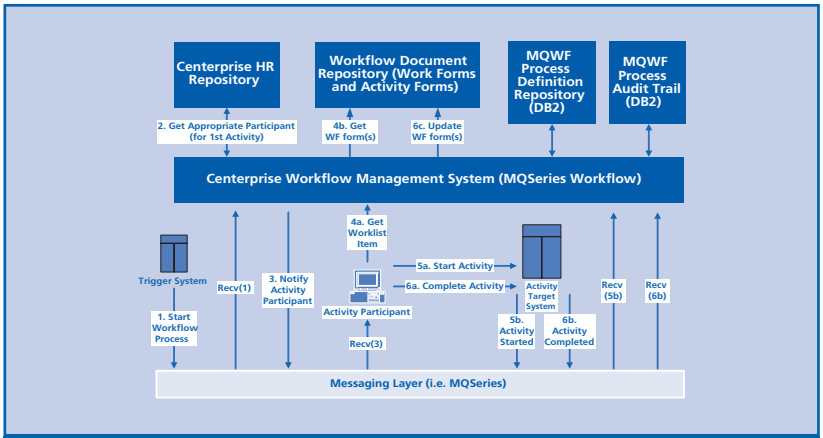


Figure 2.2: Work flow architecture

Our confidence has been vindicated. In addition, some of the actions that IBM has subsequently taken with WebSphere have really assisted us. For example, the merging of MQSeries, which is omnipresent in the finance sector, MQSeries Integrator (MQSI, the broker) and MQSeries Workflow (MQSW) into the overall WebSphere middleware portfolio has made many development and infrastructure aspects simpler for us.

Today, all these middleware products are used in CenterSphere, depending on which component(s) you purchase. In most, you will have WAS and MQSeries; in others, you'll find MQSI and MQSW as well.

For example, we use MQSeries as the decoupled messaging backbone between components. Yet we also use a Web Services-type approach for some forms of fast local processing, usually within a module. In effect, by going down the WebSphere route we have not had to worry about building basic infrastructure middleware. It is provided for us, by IBM.

This has enabled us to move up the processing stack. We have come nearer to the process and work flow dimensions. In so doing, the range of WebSphere capabilities becomes clearer, as does its attraction.

We found (and remember that we are speaking about a timeframe of two, to two and a half, years ago) that IBM already had a sense of how messaging could facilitate and enhance both process and work flow. This was different than with any other vendor we talked to. As a result, we have been able to concentrate on building customer-relevant functionality rather than on having to create our own middleware or infrastructure.

Process and work flow considerations

In the qualitative risk management context, work flow and process flow take on new meaning. Work flow and process flow are not just routers of activities from person to person or application. Instead, these become a key aspect of automating the qualitative dimension.

What we have done is to extend and deploy work flow (Figures 2.2-2.4) as a control mechanism within complex financial organizations, which have many complex processes and activities. We have used and then expanded on MQSW to underpin the financial services working environment.

For example, in the case of OrganizationCenter, we built upon MQSW to provide work management and reporting. This enables our customers to identify and then describe their reporting lines and responsibilities within their organizations. While this sounds simple, it is not. Describing chains of command and capturing them so that they are continually accurate is a particular challenge. It is worth little if management reporting lines, roles and accountability are not included. In turn, this means including information about how an organization's different entities are related to each other as well as how each individual's responsibilities are directed.

Here is an example. A market risk manager can have several different sets of responsibilities placed on him or her. In addition, there can be a different set of business standards by which he or she will need to execute that responsibility. So far, much of this is information-based, although many firms have not yet gotten as far as systematizing these. But this is not sufficient for exercising day-to-day operational control where management or regulators want to see what are the authorized practices for a particular individual at any one time and whether these have been observed.

To deliver this, we had to go outside of the conventional work flow processing paradigm, which all too often is associated with paper flows within an organization. We worked

closely with IBM's Boeblingen Lab on a design point that logically gives us a user exit at name resolution time. This enables us to move beyond what, in our view, is a current limitation of most work flow products — that they only really allow you to possess one organizational reporting line structure. While this may work well in non-financial organizations, financial sector reality has people with 'dotted line' responsibilities stretching all over the place, and often between different legal and operational entities.

In traditional work flow products, it is extremely difficult if not impossible to have multiple reporting lines — such as geographic, product and functional structures — all operating in parallel and working together. We have overcome this with our own object-relationship model and by being able to exit from the work flow engine to access specific functions that we have designed and built to accommodate real-world realities.

The other aspect of work flow that led us towards the WebSphere suite of middleware is that it, MQSW, includes a powerful audit capability within the MQSW work flow engine. Given the kinds of records and regulatory requirements associated with risk management control — where people want to go back in time to find out who did what to whom and when — not having to build a credible audit trail through work flow processes saves us on development as well as making it much easier for us to provide audit information easily and quickly.

Indeed, we think we have done neat things to take advantage of this MQSW capability. Whereas most workflow users seem to think of it, work flow, as a straight line, automation or repetition of identical activities, such as those found in an insurance claim or on a manufacturing shop floor, we use work flow to ensure that corporate executives, for example, are fulfilling their fiduciary responsibilities — and we can prove that they have done what they should.

All of a sudden, a CEO and a financial firm's auditors or regulators perk up when we make a presentation. They relalize they can automate what would otherwise have been accomplished with a mix of ad hoc paper, pencil and goodwill.

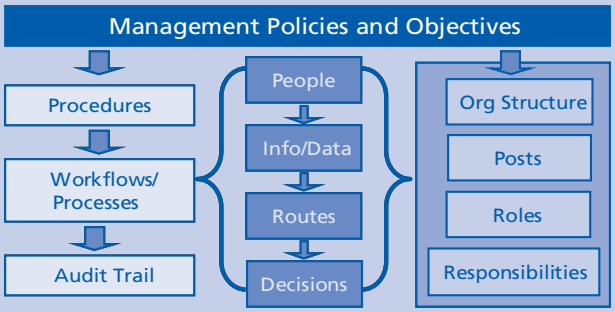
This has a financial impact. If you can show regulators what is happening, and that procedures are being followed, you can free up capital for other purposes. This will particularly apply when the Basel II Accord is implemented, where there is going to be greater emphasis on self-policing. If financial organizations can show they know what is happening, the requirements for capital might be reduced, thereby offering competitive advantage in the marketplace.

Tieing together information, tasks, roles and people

The ability to tie together task, role and people is, we believe, a true differentiator for us at Centerprise. We deliver it because we make use of MQS, MQSI, MQSW and WAS functions, albeit under the covers, as far as the user is concerned.

We use combinations of these products to obtain and deliver information. This can also be broken down in several different ways. We find that we use MQSI to handle data components, the data feeds and data transformations. It also serves as a key part of assembling the trail about what data arrives where, when did data get transported, to whom was it sent and in what form.

Figure 2.3: Management framework



In contrast, MQSW is much more oriented around people and tasks and roles. It is what manages these and provides the people-facing audit trail. In contrast, MQSI does this for data.

When you purchase our modules, this is all brought together. If, for example, you look at an end to end process you may have legacy-sourced data feeding a P&L process. But once the data has been transformed and you run the financial analytics, that P&L report can be sent out in a work flow managed way to the appropriate executives who have to review that report. The work flow engine can monitor and indicate

whether they have reviewed the information and record what actions were taken or need to be taken. Furthermore, this can be on a daily, or any other, periodic basis (see also, Figure 2.5).

In a current client engagement, we are focusing on workflow as a tool to improve the efficiency of mostly manual processes, especially when coupled with some of our integrated functional modules. As work flow becomes more pervasive, we are expecting this particular client to be able to support several of the enterprise-wide improvement initiatives in which they have interest, including activity-based costing, Six Sigma analysis and operational risk mitigation.

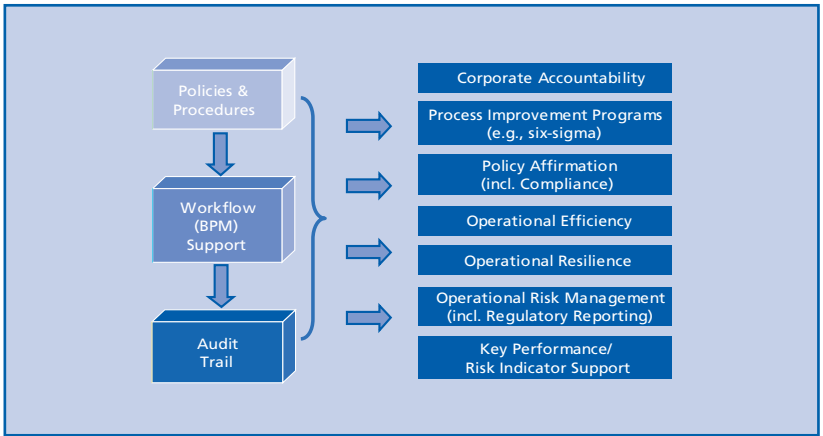


Figure 2.4: Benefits flowing from work flow

Work flow modules and layering

When it comes to deployment of CenterSphere, each module can run on its own system with its own application server. Communications between modules can be, for example, via MQSeries. Alternatively, combinations of modules may run on the same system, and will likely also use MQSeries for inter-module communication, not least because of the transaction reliability of MQSeries. There are two sorts of exceptions to this:

- **ConnectionCenter**
- **CenterFlow.**

ConnectionCenter is where information and data flow in and out. This can run on a physically separate system, although it need not. But logically it occupies a role akin to a hub between the other application modules. Much of its work is associated with:

- **data transformation**
- **format validation**
- **routing**
- **recording of what has gone where.**

It manipulates data that is in flight. In addition, it supports exception reporting and even triggering of work flow actions that must then be followed up.

Work flow is different. If the 12 application modules talk to the thirteenth module in the center, the ConnectionCenter, work flow is like a function that sits above all the other modules. It is the manager of managers, both systems and people. In this context, work flow tends to be the highest level or layer. It builds on everything else. And we are constantly finding that work flow is becoming an encompassing function, one that is needed and used by all of our other applications.

For example, if you go into one of the sub-systems that lets you look at a particular report or perform a particular operational function, what automatically appears — once you are within that function — are the appropriate work flow processes. If you are doing an HR operation in the OrganizationCenter module and you have to manipulate an employee — such as adding a new one, changing an address, increasing a salary or something similar — an appropriate work flow related to the particular authority appears at the user interface of the person who is authorized to undertake that particular task.

Similarly, if you go over to a P&L report, when you are at a user interface, you may see certain work flows that relate to a particular report.

What we have is a work flow engine that acts like a layer above the subsidiary processes in each module. Furthermore, if you need to create a new procedure, you define this as a work flow that comprises multiple processes that already exist in one or more of the other modules. This provides great flexibility as well as fast reaction or change times.

Furthermore, work flow is not only an overarching control on top of our own modules. We can apply it to existing, or legacy, processes and applications located elsewhere. Again, in the finance sector, the broad adoption of MQSeries helps here.

The challenges in making the middleware work

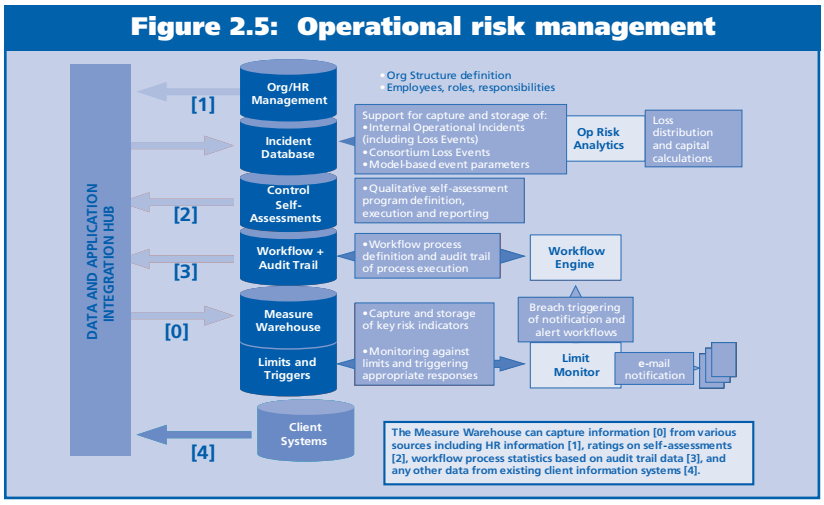
When we undertook our original selection process, there was relatively little at that time which, in functional terms, differentiated Java products. The attractive dimension about IBM, besides its close relationship with the finance sector, was that we did not have to put together the various pieces of middleware — such as messaging, broker, work flow, application server, etc. — in the way that we might have had to with other vendors. That is because we were having either to purchase from two or more vendors or because other vendors had not integrated their middleware as adeptly (as IBM).

That said, we still faced challenges. The first was the learning curve. This was quite steep. We expected that, especially across such a rich set of capabilities. The second challenge was that we found, as in the HR example described earlier, that there were areas where we had to extend IBM's middleware to obtain specific capabilities. Usually this was not a major problem, but it was not always straightforward to work out the best way forwards.

On the other hand, there were areas that were simpler. We did not think that MQSI calling MQSW to start a work flow — for example, when a process goes wrong — would be as straightforward as it turned out.

Also, IBM went through quite a lot of pain. While a swipe of the pen might announce the integration of the extended WebSphere family — when MQSeries, MQSI and MQSW were added to the WebSphere Application Server — the actual products have taken (and are still taking) time to coalesce into a family. At times, the underlying tool sets have stayed a step or two behind, which has meant we have had to work that much harder.

Figure 2.5: Operational risk management



Looking back, when we put all these points together, the main challenge was to adapt the 'standard' work flow engines to support the kind of business process methods and metrics that wholesale financial service organizations require. As we focused in on the wholesale financial sector, we understood that its business processes involve significantly more complex flows with much more complex rules about how you resolve the appropriate actions and involvement of people.

Much of our concentration has been, therefore, upon work flow. If we had had a work flow kernel that we could have customized, it might have been easier, but probably not to a huge degree.

On the application server side, we started with WAS Version 2.0. This was pretty primitive and even a little scary. It really was not until Version 4.x that IBM's offering became serious and capable of being implemented in a production environment (MQSeries, MQSI and MQSW were already production proven).

On a related topic, WAS (and indeed most of the other IBM infrastructure middleware) is not yet easy to administer. There is definitely work necessary here to match some of Microsoft's tools.

For the broker, we started using MQSI Version 2.0. However, because most of our formats were delimited in some form, we had to use the New Era of Network node capabilities. Moving to V2.1 greatly shortened our development cycle, as well as performance.

Returning to the management thought, another area is combining transaction awareness. Right now, if you are within MQSI, you have good identification of the transaction. Similarly, if you are in the MQSW work flow, you have a good idea of the scope of its transactions. But relating the two transaction aspects together, or adding in WAS-based processes, remains a problem.

We are having to build our own schema for a sort of universal identifier of actions that crosses IBM's middleware product boundaries. The attraction of this to us is that we can capture events and perform processing in a real-time way. That means that when we go into MQSI, we can generate events through application logic. Simultaneously, in MQSW, we can set a

switch that says every time something starts, create an event.

Once we have built these capabilities, we will have a system that can monitor itself in real time, including taking automated corrective actions or alerting people to take those actions. This has particular appeal from an audit and qualitative control and management perspective.

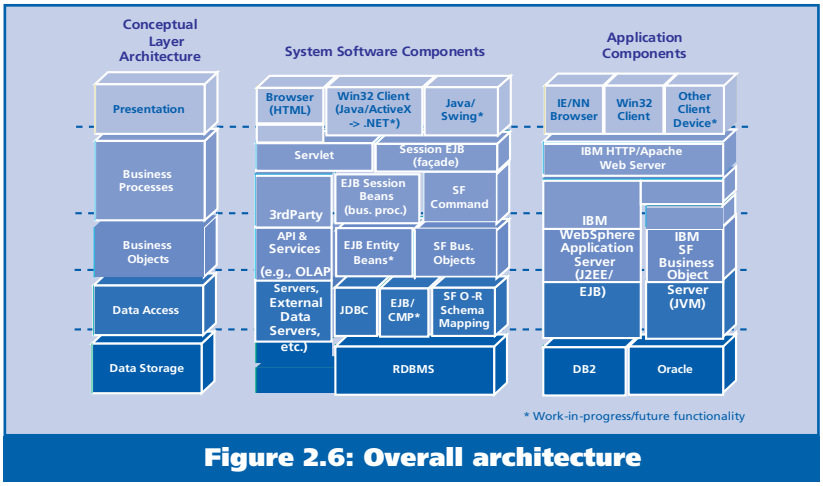


Figure 2.6: Overall architecture

Lessons learned

The first lesson learned was, and still is, a pleasant surprise. When we visit prospective customers, especially those who have already thought about distributing processing and managing by automated process and work flows, these prospects seem to have come to the same broad conclusion as we have about what an applicable multi-tier architecture should look like (Figure 2.6).

We are also finding that across the wholesale financial sector there is an appreciation for messaging-oriented processing combined with work flow for management and control purposes. Financial firms are also looking at brokers because they have understood that their multiple application systems are totally unmanageable without some form of automated broker with data transformation and routing. Similarly, the Java, JavaBeans and J2EE story has been accepted by the finance sector as the way forwards.

This has all been very good for us. Financial institutions are coming in, describing their 'ideal' architecture — and it is the one we have built. We feel our design choices are constantly being validated.

Looking from the business side, it has become clearer that work flow and business process management have become the key aspect of the CenterSphere Suite of applications. It has gone much further and with greater effect than we thought or expected. What we have learned is that it is the whole process that must be managed, and this subsumes both the qualitative and quantitative aspects of risk. Process management has become vital to successful operations. The focus on corporate accountability, post-Enron, only strengthens the argument.

The last lesson learned might be argued. We used to introduce technologies and put up with, or wrung our hands over, how complex the infrastructure, especially the middleware, was to work with. Looking back, we think we didn't sufficiently ask ourselves, 'why are we in this particular ditch?'

We think that has changed now. Middleware has improved, stabilized and, while not end user friendly, it is usable in the hands of competent developers and operations people who have understood that they do not wish to reinvent or rewrite what has been done elsewhere. This applies as much to software vendors as to customers. Both want to focus on the business problem. This is a big advance — and for the better.

Management conclusion

CenterSphere is an enormous suite of application software, with extensive capabilities. Beneath the application function are three other attractions:

- ***the relevance of the ASP technical model to create a 21st century architecture of federated systems that work over the Internet***
- ***the importance of 'packaged' middleware which can provide a production class infrastructure***
- ***the increasing significance of work flow when combined with process flow to manage both data and people processes in a way that provides an environment where qualitative as well as quantitative risk management becomes possible.***

All of these represent major steps forward, particularly when an infrastructure can be expected to be built on middleware that is available as tried-and-tested, production-ready products — rather than as sets of capabilities which the customer must link together.

Do enterprises need a meta operating system?

Amy Wohl
Principal
Wohl Associates

Management introduction

Just when you thought your biggest operating system problem was managing the migration from one generation of Windows to another — or considering an alternative operating system environment such as Linux — a buzz has started around the idea of a higher-level operating system or environment. This buzz comes from the likes of IBM (with increasing talk of 'autonomic computing'). It also comes from Sun, with its promotion of N1.

In this analysis, Amy Wohl examines how different vendors — from Sun to IBM to HP and others, and including Grid initiatives — are approaching this issue, which highly resembles middleware, albeit of a sophisticated form. In particular she looks at its relevance to user futures.

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

Nomenclature

This idea may be called a meta operating system (an operating system for operating systems). I have also heard vendors describe similar visions as:

- **systems management software**
- **network operating systems (but I do not think they have a replacement for Novell's NetWare in mind).**

The term 'Internet Operating System' has also surfaced. Nevertheless, this sounds overly grand and ambitious.

So, do not count on nomenclature to identify what we are talking about here. At this stage, the discussion is more about a vision of how to gain flexible control over increasingly larger and more complex computing environments, rather than about an agreed upon label. Indeed, it is quite possible that this is the flip side of Web Services — where everything receives the same name but, under an identical label, each vendor offers something different.

What is a meta operating system?

Whatever we eventually decide to call it, a 'meta operating system' (Figure 3.1) is designed to help IT to:

- **configure**
- **monitor**
- **manage**
- **control**

the entire computing environment. This has become an ever increasingly difficult and complex chore. You only have to examine what a computing environment can look like these days. We are, therefore:

- **no longer talking about a two-tier client/server environment, or even an n-tier one**
- **but rather a complex and dynamic amalgam of everything.**

For example, it might include:

- **PCs, workstations and terminal desktops of any vintage — running standalone applications or server-based applications, or using browsers to interface with remote systems**
- **servers of any and every description, running any operating system and with applications of many vintages (from architecture to language); often these will be attached to remote (perhaps legacy) databases**
- **mobile devices of any description, in any location, and all with an expectation that they will be able to connect securely to designated systems where these devices can participate as if they were ordinary desktops located within a serviced location**
- **participants or users of IT who are not only physically remote but who are also not entirely controlled by the IT department; this may include contractors, suppliers, partners and customers as well as an organization's own staff.**

As if this was not sufficiently stretching, there is an increasing pressure to farm out parts of the IT infrastructure (including hardware, infrastructure and even applications) to less expensive, more efficient (but physically remote) business partners. These can include:

- **ASPs**
- **outsourcers**
- **hosting providers**
- **new service propositions, like the IBM's Information Utility, 'e-Business on Demand'.**

Moreover — and this point is at least as important, and perhaps more important, than the support of complexity and heterogeneity — we need to do this with more reliance on policy driven systems and automation, rather than human hands. There is just too much to do.

IT drivers and benefits

IT management, within user organizations, is, of course, interested in schemes that will save their increasingly pressured budgets time and money. The notion of systems management is not a new one. But, so far, it has been mainly about trouble alerts and avoidance in search of better performance and higher quality of service.

Unfortunately, the approaches are almost invariably piecemeal. No vendor has yet managed to include everything. Much is, at best, semi-automated. Put bluntly, most systems management can be described as a 'scheme for exception reporting, with a human and his or her pager at the other end'.

The idea of a meta operating system is, therefore, to:

- **move further up the hardware and software stack**
- **try to include 'everything' (but take that with a grain of salt)**

- **provide as much automation at any point in time as computing power, policy rules and IT's need for control will allow.**

The players and the plays

The players are gathering. I first took note of a new round of this game last February when Sun started to talk about its N1 project at its annual analysts briefing. Six months later, it is still possibly the best developed idea, which is why I will concentrate the most space on it here.

But to think that Sun is alone is inaccurate. In fact, nearly every major system and infrastructure software vendor has an approach to this idea (announced, unannounced or in the works). In some cases it will, like Sun's N1, be introduced as something new, daring, unique and imperative. In other cases it will be offered as an evolution of an existing scheme (systems management) or the natural growth of a relatively new idea — such as storage virtualization, autonomic computing or even Grid computing.

The players with the best chances are — as usual:

- **either those with the resources and market power to establish broad, cross-platform de facto standards (this time, of course, based on open standards and open APIs)**
- **or, alternatively, a major player with a very strong partner network of equally influential partners.**

Keep the cross-platform thought in mind. Several of the strategies I have looked at are definitely limited by their desire to push their own platforms and ignore or treat as stepchildren some or all of the competition.

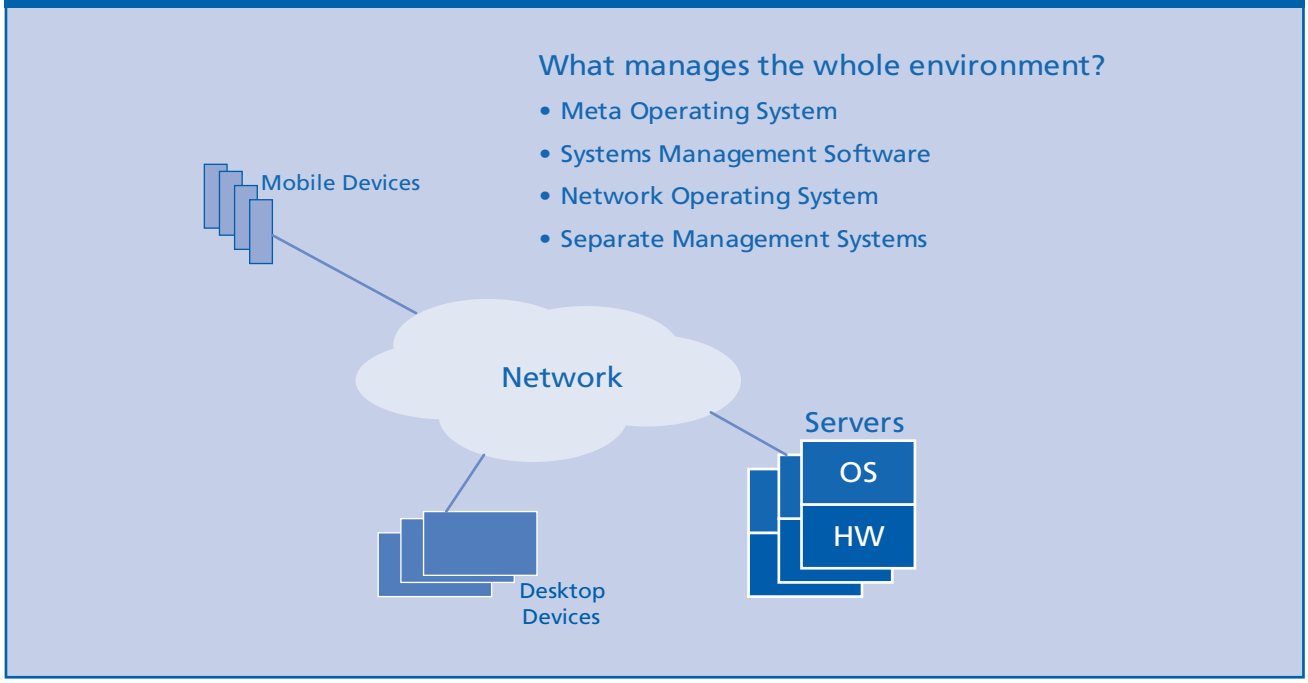
Sun's N1

Sun sees computing needs today as being driven by a desire to reduce costs while maintaining flexibility and high availability. For Sun, a turn to infrastructure software is a way to maintain both revenue and market share.

Since Sun announced N1 in February it keeps changing its mind about just what this vision is about and what it would like us to call it. Originally described as a 'meta operating system', Sun now prefers to describe N1 as a scheme for managing Operations via a Software Service Plane. In effect, it likes us to think of it as a method for virtualizing all of the computing environments' resources — CPU power, storage, networking, etc. — and then assigning resources and managing these according to their importance to the organization.

As such, N1 should be a combination of storage manage-

Figure 3.1: Meta operating system (conceptually)



ment, systems management, network management and some operating systems functions, all based around:

- **understanding more about applications than these functions have in the past**
- **how to operate applications in a highly heterogeneous environment.**

Physically N1 is software running on multiple servers. This control plane must be physically connected to the server console and to the network. Merrill Lynch has referred to N1 as a blade operating system (and with the coming popularity of blade architectures, this may yet be an important play in that market).

The new architecture is approached by a combination of disaggregation and recombination, which will:

- **be achieved by Web Services**
- **require a new level of high-level systems management.**

The idea is to disaggregate what normally occurs on the server (server state, function and computation), storage, network and software — and to then recompose it from:

- **optimized entities**
- **appliances**
- **intelligent storage**
- **compute engines**
- **EJBs**
- **software services**
- **load balancers.**

When the management system disaggregates — in order to optimize its separate parts — it makes it harder to do security and management. Since there may be hundreds of servers to manage (think of the new blade architectures coming), administrative efficiency becomes more difficult to achieve and, in fact, the correct measure is going to be 'services' (rather than servers). This only makes the administrative task both larger and much more difficult.

Sun wants to provide an administrative or operations view of the computing stack which reduces complexity and increases productivity. This is akin to abstracting the programming layer, in the way that EJBs are intended to reduce complexity and increase productivity for programmers.

N1 is intended to provide this abstraction layer, for administration. Today, most application servers deliver about a 20% utilization rate; in a well-administered environment it should be possible to achieve an 80% utilization rate. Because Internet usage is unpredictable, customers over-provision — as a form of performance insurance. This means utilization rates may actually be as low as 10%

Sun believes that an operational environment, like N1, will not only lower administrative costs and the cost of operations by allowing administrators to manage at a higher level of abstraction, it will also:

- **unify vertical and horizontal systems, including SMP, blade and high availability clusters**
- **combine compute, storage and network elements**
- **deliver end to end high availability within the network**
- **provide automated dynamic provisioning and management.**

N1 and managing services

As such, N1 focuses on managing the services provided by the elements of the network, ranging from:

- **managing end to end service levels**
- **managing the network within each service point to provisioning the physical infrastructure — including the network, provisioning and management of changes to the software stack**
- **providing automatic and dynamic resource allocation.**

Think of N1 as a fully virtualized and self-managing environment, which can cope with high degrees of heterogeneity in any dimension. Sun, however, seems less ambitious than some descriptions of Microsoft — N1 is expected to manage to the router, but not into the Internet cloud.

N1 does this via a Control Plane that virtualizes network, compute and storage elements. It is software running on a series of virtual servers that contains information on all the services (applications and data sources). This meta information will be collected as an XML tag and used by N1 to build application stacks, define services, and provision resources. N1 will submit services to the computing resources, maintain service levels and provide integrated billing.

As services are directed to virtualized computing resources (vertical SMP domains and horizontal simple blades):

- **virtualized storage is provisioned (including fabric, intelligent storage subsystems and shared file systems)**
- **virtualized network resources (soft cabled switches, intelligent elements, firewalls and load balancers) are assigned.**

In this intelligently and automatically managed environment, 'services' are rearranged by load, based on business priorities. Because the 'service plane' can move a 'service' from one computing environment to another, service levels can be maintained with high availability and little chance of failure.

N1 leverages the breadth of Sun's existing and new technologies. It is intended to align Sun technologies for the future, offering an increasing span of management and control over ever larger networks. But Sun does not want to compete with Cisco, hence its intention to stop at the router.

Where does N1 start?

Of course, all this has to start somewhere. Sun will start with vertical systems within enterprises. N1 is both a separate product that gets charged for separately and a set of components that gets built into new and existing products. The control point is a separate, separately charged for product. In the future, hardware and software products. will be sold as 'N1-ready'.

Sun will offer N1 for Solaris, Linux and Windows environments (with hardware support only for Windows). Sun has no current plans to support other enterprise UNIXes — such as HP-UX or AIX — but it will ship an N1 SDK so that others can develop adapters for these environments, if these seem economically interesting.

On the storage side, Sun will support EMC and Hitachi. The SDK will have to suffice beyond that. Sun is still not sure how it will handle the switch fabric issue, where few vendors interoperate today,

Sun itself thinks N1 is more like autonomic computing than systems management. It referred me to IBM, and its eLiza initiative, when I inquired 'whether there were any competitors doing something like this?'

Furthermore, Sun sees N1 as an ever-evolving concept, without a particular end point. It expects it (N1) to take ten or fifteen years to reach full functionality when it would manage network performance, devices, applications running on the devices and the system as a whole (load balancing, assignment of storage and processing, virtualization, etc.).

N1 will likely arrive probably in several stages (Sun currently thinks three). One reason for these stages is that Sun feels it needs to find out how humans want to work with the system and then to adapt.

Sun says it has been working on the N1 project (or its predecessors) for about 18 months. There are some pieces, including some virtualization support, in Solaris 9. Roadmap announcements will come later and future announcements will focus will on the heterogeneous and network-centric.

Storage vendors

Besides Sun, I have also discovered that nearly every major storage vendor and storage management software vendor has some kind of scheme in progress to extend storage virtualization into a more extensive scheme. While these might not be quite as ambitious as Sun's N1, they certainly move in the same broad direction.

At the same time, it note that many storage hardware vendors (who previously partnered with storage management software vendors for their software) are starting to bring their own storage software to market. Some of the big storage management software vendors seem unconcerned — they clearly feel their knowledge is too far ahead of their partners for this to be a serious threat. But if the real issue is to build more integrated products — that go beyond storage management systems — vendors like IBM, HP and Hitachi (all of whom sell storage as a major offering) could become significant players. For example:

- **Hitachi has recently unveiled its own storage management software (High Command)**
- **IBM and HP already have both Storage Management and Systems Management software offerings of their own.**

Systems management vendors

There is always a demand for systems management. Successful vendors include:

- **Computer Associates (with Unicenter)**
- **IBM (with Tivoli)**
- **HP (with OpenView).**

While none of these products have quite the scale or breadth of scope of what is contemplated in Sun's N1, they

could be evolved to take on a larger role if their architectural design proved suitable not just for the complex, heterogeneous environments (which they anticipate), but also for the idea that these environments:

- **may exist in many disparate locations**
- **under the control of many different systems and organizations.**

The real issue here is that these systems management vendors possess:

- **already recognized systems management products (which Sun does not)**
- **strong and inclusive cross-platform strategies (Sun stops at Solaris, Linux and Windows, leaving the rest to what partners might choose to do).**

I particularly like the OpenView play since HP and Compaq will combine their resources around the OpenView product which is already popular and used broadly (including in competitive environments). HP intends to port OpenView to the Web Services environment. If it can accomplish this quickly that could give the 'new HP' a running start to leading the market.

Recent additions to OpenView have included a storage management module and a Help Desk, together with an approach that permits for more automation (the 'new HP' calls this 'intelligent correlation'). These are all positive steps in the right direction.

Web Services environment management vendors

I suspect that, while lots of smaller vendors may have partial solutions in this space, the next big play will be an attempt by Microsoft to assert control through its .NET strategy. In writing about the next Open Source Conference, industry pundit Tim O'Reilly said recently: "With .NET, they (Microsoft) have a new vision, of an Internet-scale operating system, which is driving the industry forward into uncharted territory."

O'Reilly then goes on to express concerns about Microsoft's various aggressive attempts in the past to maintain control and revenue growth. But I think he is exactly right.

Microsoft will try to create a new scale of operating system where the desktop and the corporate server are merely pieces of some much larger operating system environment which is under Microsoft's control, in the sense that Microsoft will:

- **continually change functionality through upgrades**

Figure 3.2: IBM's phases of progression

Basic	Managed	Predictive	Adaptive	Autonomic
Monitoring products through individual consoles	Consolidation of consoles for monitoring	Role-based consoles with analysis and recommendations	Self-management based on resource policies	System self-management based on business policies
Single product manual installs	Auto product installs	Auto install with recommended parameters	Auto product installs Auto install and self-configuration with dependency resolution. Auto discovery of resources. Virtualized resources with automatic provisioning	Auto install, config. Virtualized resources dynamically provisioned to address business objectives
Products providing monitoring data in disparate formats Humans react to problems and resolve manually	Correlation and filtering of some events. Hard coded responses to resolve common problems	Common PD data formats. Correlation and analysis of events Recommended responses presented to humans to act on	Actions taken to heal the element based on information it has	Actions taken to heal the overall system in concert with business policies
Manual processes for reviewing measurements and provisioning resources	Load balancing, deployment of new software images	Parameter recommendations, real time mining of performance data with analysis and recommendations	Automatic provisioning of additional resources within a pool	Dynamic provisioning of system resources to meet business objectives of the solution
Point solutions for security	Corelation and filtering of security events	Analysis and recommendations to prevent security breaches	Self-protecting actions taken	Systems-wide self-protecting actions taken

- **permit content to be transmitted only if it meets its rules of Digital Rights.**

You do not like this? It is time to either unplug from Microsoft (unlikely) or start lobbying your legislator to prevent this from occurring.

Of course, Microsoft itself has its limits. It is unlikely to support anything beyond its Windows environment, so companies that choose other operating systems will simply exist in a separate, parallel universe. That, of course, is not good for systems management or interoperability.

Autonomic computing

Simultaneously, IBM has been working on a series of initiatives to make computers better able to manage themselves. This starts with individual computers (eLiza) and moves up into large and more (much more) complex systems with autonomic computing.

To date, this is a set of intellectual work-in-progress and some resulting technologies which are gradually being embedded in products, rather than a product itself. You can read more about IBM's efforts to apply the paradigm of the self-regulating human nervous system to the problem of ever larger and more complex systems at http://www.research.ibm.com/autonomic/.

IBM's Autonomic Computing Strategy was described by the IBM Software Group's VP and General Manager, Steve Mills, on September 11, 2002. He showed (Figure 3.2) an evolving set of increasingly automated function which eventually permits systems to be self-configuring, self-optimizing, self-healing and self-protecting.

Beyond describing autonomic computing as an evolving technology, and frequently noting that it is a technology that will be embedded in many products, rather than a separate product offering, IBM has not offered specific time frames. Some autonomic computing features are, however, already available in shipping IBM products while others are clearly for the future.

In Figure 3.2, note how ambitious and specific IBM's plan already is, with functionality becoming increasingly automated (and less manual or human-intensive) as you move from left to right across the chart on any level.

Shortly thereafter, on September 18, 2002, Sun's Steve MacKay, Vice President, N1 and Management Systems described a Roadmap (Figure 3.3) for its N1 product offering. This is described as being in three phases.

You will note that many of the goals of IBM's Autonomic computing initiative and Sun's N1 are similar, but that IBM has very specific objectives in each of its five phases while

Figure 3.3: Sun's phases of progression

Phase I Virtualization 2002-2003	Phase 2 Services Provisioning 2003-2004	Phase 3 Policy Automation 2004-2005
Customers will be able to use N1 technologies to transform individual computers, network elements and storage systems into an aggregated pool of resources	Customers will be able to use N1 to specify the business service definition for a Web Service	N1 will include telemetry and metering capabilities
Once virtualized, the system will be able to allocate, monitor and meter the usage of included resources	N1 will take care of provisioning the data center resources required	N1 will enable the automation of work required to maintain application service level objectives
		There will be more automated policy generation, automation ease of use and extensability.

Sun's goals remain more general and less defined. I suspect that this is because:

- **Sun is still in the process of determining exactly what level of functionality will actually be available in a given Phase**
- **IBM intends to offer additional specific function and automation, as it moves from left to right across its chart.**

Grid computing

IBM is also an important player in Grid computing, an ongoing effort to permit computers in multiple locations to share their available processing power and to together create a very large computing resource. If you think that sounds like a kind of CPU virtualization, of course it is.

Grid computing started out as a way for scientific researchers, especially in academe, to run compute-intensive calculations. But it has spread, and is spreading, to many possible applications and platforms.

Other vendors are also offering their own Grid initiatives, including:

- **Sun's Grid Computing offer**
- **a Planetary Computing initiative from HP.**

Recently, IBM has noted that Web Services are particularly well suited to the Grid Computing environment. It suggests that the Grid can even become the operating system for a network-based Web Services environment. In other words, the Grid is just another route to the same destination.

Management conclusion

More complex, much larger systems — enabled by wide area networks incorporating the Internet — require ever increasing automated management techniques. Every vendor that supplies part of the infrastructure — hardware or software — is now considering how to be part or all of the solution.

Ms. Wohl expects the serious choices to come from the biggest systems vendors, especially those who understand that they must support a broadly heterogeneous assortment of operating systems, networks and devices — not just their own and their preferred partners. But it will also take vendors with enormous resources and great customer credibility in service and support to deliver — since these are complex undertakings.

It will also require a significant effort to convince IT to permit a computer do some of its (IT's) work, no matter how badly it (IT) may need this capability to take some of the work load off its own operations.

Emerging Internet middleware

Dr Keith Jones
IBM Software Solutions Worldwide

Management introduction

In recent years middleware has become increasingly standards-based as it has addressed the needs of Internet-facing application systems. A new generation of industry standards is now emerging which is focused upon realizing collaboration within and between enterprises at electronic speeds. This is giving life to emerging vertical or 'Internet middleware' technologies and products.

In this analysis, Keith Jones reviews a selection of:

- ***the most important outstanding issues facing early adopters***
- ***the more recent standards proposals made by leading vendors to resolve those issues.***

The Internet middleware stack is not yet complete. But already it is clear that emerging Internet middleware will play an important role in the strategic timeframe for many enterprises. The key question is: how soon will your enterprise be able to capitalize on the benefits?

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

Internet middleware

As enterprises evolve to meet the challenge of competing in today's marketplace, the need for a cost-effective infrastructure that can accommodate collaborations between applications obtained from widely different sources is becoming critical to success. Increasingly, those sources include other enterprises — acting as suppliers, bankers or providers of critical business services. Such collaborations facilitate both integration:

- **within an enterprise (intra-enterprise)**
- **between enterprises (inter-enterprise) — effectively, in partnership.**

Most enterprises have already implemented or acquired application systems written in different programming languages running on a variety of middleware systems, different operating systems and different vendor platforms. Building new applications and integrating old ones in such heterogeneous scenarios is becoming increasingly expensive, and yet this is critical to competitive success.

Industry standard Internet technologies now running on most operating systems and vendor platforms have so far enabled communications between human end users and computers running the World Wide Web and simple mail applications (Figure 4.1). This has prompted many to imagine that those same technologies, with extensions, could underpin interactions between business applications independent of their location, language, middleware, operating system or supplier.

Web Services standards (UDDI, WSDL and SOAP) and technologies have been proposed as those extensions for this purpose. These are complementary to:

- **existing Internet communications protocols (HTTP, SMTP)**
- **the XML data format description and related technologies.**

They (Web Services) revolve around the concept of business 'services' being accessible via Internet communications. They envision data messages flowing along and using standard interchangeable XML formats.

In essence, the Web Services vision predicts that, over time, enterprises will consume each other's business services via program to program communications using these Internet technologies. The result that is hoped for is the realization of cost savings as well as the delivery of additional revenue opportunities — both being obtained from highly efficient electronic partnerships.

Initial Web Services technologies are already beginning to demonstrate this potential in early adopter scenarios. While some organizations are focusing on external business partnerships, most are applying Web Services technologies to intra-enterprise integration projects. But what are the critical success factors and what remains to be done to bring the benefits of the service approach to the larger majority?

The services approach

In a Web Services world every service has a description written in the standard Web Services Description Language (WSDL) that is available:

- **either through Internet directory look-up**
- **or by direct access at enterprise Web sites**
- **or from other sources (via Intranet or similar, private, sources).**

Already a 'stack' of technologies that utilize such service descriptions is available for early users to build and exploit business application services. But the concept of application or business services is not new. Using proprietary technologies, some enterprises have already built systems that possess a service-oriented architecture with success.

The value of such an approach lies in the decoupling of service providers from service consumers as well as the potential for re-use of well-tested, robust service application code. New services can be introduced without existing user code having to be changed. Equally, new users can be introduced without existing service code having to be changed. If this separation from change is extremely valuable within enterprise systems, it is absolutely essential for external, inter-business, partnership scenarios.

Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP) standards have enabled early adoption of the service-oriented approach across vendor platforms. Web Services technologies are becoming the foundation for service infrastructures that incorporate widely different middleware and operating system platforms, because a growing number of vendors are making implementations of these standards available (Figure 4.2).

However, the success of the service-oriented approach depends upon the strength of the service description and its deployment in real world systems. Details of the service interface, its operations and message formats and the service addressing and protocols supported are contained in the service description. If those details are well described, and relatively stable, they form the basis for a contract between run-time service providers and service consumers.

The inclusion of Internet communications, SOAP exchange protocols, XML data formats and WSDL service descriptions into a growing number of systems has prompted some to think in terms of a standards-based vertical — or 'Internet Middleware' stack (Figure 4.3) — for widely distributed application systems. Such a stack is now emerging as technologists tackle the complex problems of integration and as vendors make implementations available.

Service reliability

Some early users of Web Services have discovered that Internet protocols, such as HTTP and SMTP, are not reliable enough for certain business scenarios. When it is important to know that a particular interaction between applications really happened once, and only once, a more reliable messaging transport is required.

SOAP messages flowing over HTTP can be made more reliable by adding application function. But a more robust solution would be to have those same messages flowing over a more reliable underlying Internet protocol. This is an acute issue in an inter-enterprise environment, with different legal participants. To address this IBM has proposed HTTPR as a standard for this purpose. Several implementations have already been made available but there is no agreement as yet on what the long-term industry solution should be.

Service access reliability within enterprises (intra-enterprise) is not an issue of such magnitude. When service providers and consumers support reliable messaging, for example by connecting into an existing MQSeries corporate backbone, SOAP messages can be exchanged with greater reliability. IBM has already made SOAP over JMS (MQSeries) technology available for early adopters within enterprise networks.

Service security

In addition to reliability issues related to transport of service messages, users are also concerned about security of the same delivery processes. Confidentiality is often extremely important to businesses contemplating the use of Web Services for implementation of their inter-enterprise partnerships. Similarly, integrity of messages exchanged is also of the highest concern. Forging and tampering with message content could have catastrophic business consequences, if the necessary safeguards are not in place.

Several aspects of security must each be implemented to provide a solution for Web Services (Figure 4.4). This has been recognized in the Web Services Security Architecture and Roadmap published by IBM, Microsoft and Verisign in April 2002. An initial specification for a Web Services Security (WS-Security) standard was proposed at the same time — and initial implementations were made available. Confidentiality and integrity support for service interactions and messages are included in these early implementations.

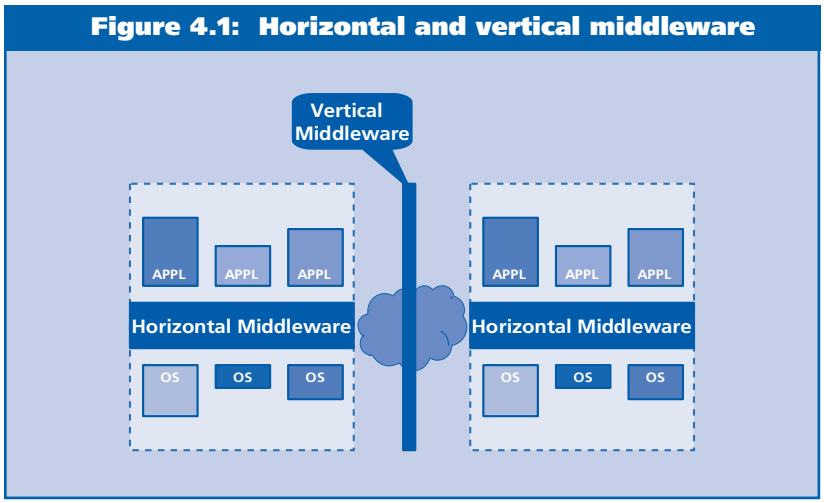
Other security concerns will be addressed in coming months. Complementary standards proposals and a growing library of vendor implementations are in the peipline.

Service co-ordination

The SOAP standard facilitates different patterns of interaction that service providers and consumers may use. Each pattern is modeled as a sequence of one-way messages. There are four such basic one-way or two-way patterns:

- **a one-way request,**
- **a two-way request-response**
- **a two-way solicit-response**
- **a one-way notification.**

Figure 4.1: Horizontal and vertical middleware



But many more complicated interactions may be composed from compound uses of these basic patterns, and that is just between point participants. This raises the question: which component in a service infrastructure co-ordinates service requests and responses as they are exchanged between multiple participants in a business scenario?

IBM and Microsoft have proposed a new standard called Web Services Coordination (WS-Coordination) to facilitate the development of new middleware that will provide necessary service infrastructure for co-ordination in the future. In their view,

co-ordination is best provided by standards-based infrastructure — rather than having many service developers develop and provide their own solutions.

The approach taken proposes that business services register with infrastructure co-ordination services (Figure 4.5) for well-defined and standard co-ordination protocols. Multiple co-ordination services may be involved in any given scenario. Thus co-ordination contexts flow with the service messages — in order to effect communications between all the participants in any given scenario.

Co-ordination may take many forms. The standard proposed, therefore, is intentionally extensible to include a wide variety of protocols. Some basic co-ordination protocols have already been proposed. Many more are expected in future. The good news is that co-ordination services, protocols and contexts are all specified using standard XML grammar. In many cases this builds upon the extensibility of the SOAP and WSDL standards.

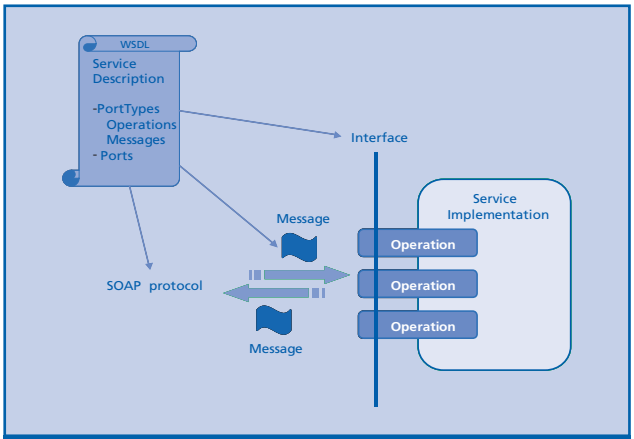


Figure 4.2: The service approach

Service transactions

Correlation of responses with requests is probably the simplest form of co-ordination expected in a Web Service scenario. However, for business services, integrity and repeatability are important qualities of service. These can be provided by transactional guarantees delivered by an appropriate service infrastructure.

IBM, Microsoft and BEA have proposed a standard for Web Services Transactions (WS-Transactions) to facilitate the development of new middleware to provide the necessary service infrastructure for transactions in future. Once again this function could be incorporated into business services via individual application efforts. Again, the general feeling is that this is best delivered via standards-based infrastructure and in a generic fashion. The approach taken has been to specify two distinct co-ordination protocols that can be integrated into the framework proposed by the WS-Co-ordination standard:

- **the first recognizes that in certain localized service scenarios, ACID transaction guarantees will be needed: the Atomic Transaction co-ordination protocol delivers this to participants when needed (Figure 4.6)**
- **the second recognizes that, in wider and more complex scenarios, the ACID approach is neither affordable nor appropriate and a rather looser business activity co-ordination protocol is more appropriate; this provides for long running transactions that may span days and**

Figure 4.3: the Internet middleware stack

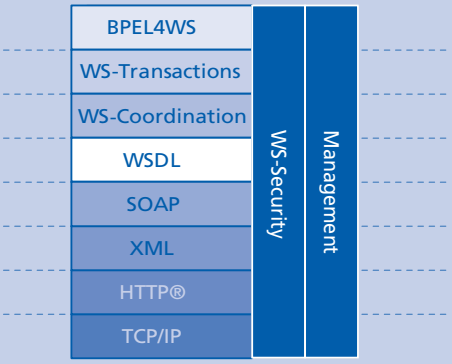
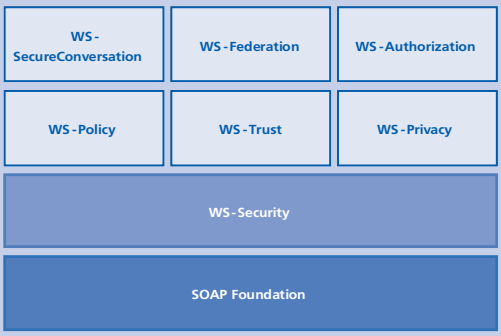


Figure 4.4: The service security stack



incorporate compensation activities when inconsistencies are discovered.

Once again an extensible approach has been taken to providing transaction support for services. This is extremely important as it allows for existing vendor-specific and proprietary transaction systems to participate in future service scenarios. A standards-based service infrastructure will provide the framework for widely different applications and platforms to participate with integrity as equal partners.

Service flows

At the highest level of the Internet middleware stack is a layer that specifies standards for business processes implemented across widely distributed heterogeneous platforms. The vision here is that elemental business services can be composed of existing complex business processes, using an array of flow and other constructs. These complex business processes may in turn be presented as business services which business partners (intra-enterprise or inter-enterprise) can invoke.

IBM and Microsoft have combined previous initiatives (WSFL and XLANG) to form a joint proposal for a standard Business Process Execution Language for Web Services (BPEL4WS). The approach taken allows for both abstract business protocols and concrete business processes to be described using standard XML for interpretation by machine.

A BPEL4WS process is basically a flowchart-like expression of an algorithm. Each step in the process is called an activity. There are a number of primitive types:

- **invoking an operation on a Web service (<invoke>)**
- **waiting for a message to arrive from an external source (<receive>)**
- **generating the response of an input/output operation (<reply>)**
- **waiting for some time to elapse (<wait>)**
- **copying data from one container to another (<assign>)**
- **indicating that something went wrong (<throw>)**
- **terminating the entire service process (<terminate>)**
- **doing nothing (<empty>).**

These primitive activities can be combined into more complex algorithms using any of the structured types provided in the language. These are the ability to:

- **define an ordered sequence of steps (<sequence>)**
- **branch using the case-statement approach (<switch>**
- **define a loop (<while>)**
- **execute one of several alternative paths (<pick>)**
- **indicate that a collection of steps should be executed in parallel (<flow>).**

Within those activities executing in parallel, execution order constraints can be specified by using links. BPEL4WS also allows for:

- **recursive combination of structured activities to express arbitrarily complex algorithms that represent the implementation of a service**
- **activities to be grouped into scopes that identify integrity and recovery boundaries using atomic and business activity co-ordination protocols.**

In the real world, business processes involve interactions with external partners. BPEL4WS allows for this by providing the facilities to define partners and the roles they play in relationships called service links (Figure 4.7). Each role, therefore, identifies Web Service interfaces that must be available at execution time. This outward-facing aspect of business process definition enables the specification of abstract components of service descriptions while leaving the concrete details for resolution at execution time.

Implementations of the proposed BPEL4WS standard are already available for early adopters from IBM and others. Nevertheless, more work is needed to expand the scope of activities that can be described and additional protocols needed to model real world business processes. That said, BPEL4WS is a bold step in the direction of massively integrating electronic businesses.

Management conclusion

A new generation of middleware technologies is beginning to emerge. So-called vertical, or Internet middleware, technologies focus on solving the problems of interoperability and integration across language, operating system and hardware platform boundaries. The vision has been described. The beginnings of an implementation for that vision can now be seen.

The underlying technologies for Internet middleware are the familiar industry standard Internet protocols — IP, TCP, HTTP, SMTP and others. Support for Web Services stan-

dards — like SOAP, WSDL and UDDI — based on XML message formats is layered upon Internet protocols to provide platform neutral communications between applications. These are then viewed as business services. Already Open Source organizations as well as leading software vendors have Web Services tools and technologies available for early adopter projects.

In recent months a new layer of standards has been proposed and initial technologies made available. These provide security for Web Services messages and interactions, co-ordination for sets of interactions that must be handled as discrete business activities and transactional integrity for both short and long-running activities that implement business services. WS-Security, WS-Coordination and WS-Transactions are not yet agreed as industry standards — but they are garnering interest and support amongst vendors and potential users alike.

Building upon this layer is yet another proposed standard, BPEL4WS, and initial technologies are available for use by early users. It provides for orchestration of Web Services as co-ordinated activities which map to business protocols and processes of arbitrary complexity. The Business Process Execution Language allows business service implementations to be described using an XML grammar that is similar and complementary to that used for Web Services descriptions.

The result is intended to be accessible to tools for compilation and to run time engines for dynamic interpretation. Middleware for integration in an Internet context is, therefore, making great strides forward.

In turn, these new layers of standards will facilitate the development of new and enhanced middleware products in the coming months as they are refined and agreed by the wider community of vendors and users. Perhaps more significant is the realization that Internet middleware could dramatically reduce the cost of integration both within and between enterprises and dramatically reduce time-to-market for some types of software, as service re-use becomes a realistic possibility.

But there is more to be done. Before Web Services can be deployed on a large scale there must be a management framework that delivers necessary controls for licensing, usage, billing, versioning and delivery to contracted levels of service performance. Research is currently underway on these topics and, in the near future,the next layer of Internet middleware will likely be defined.

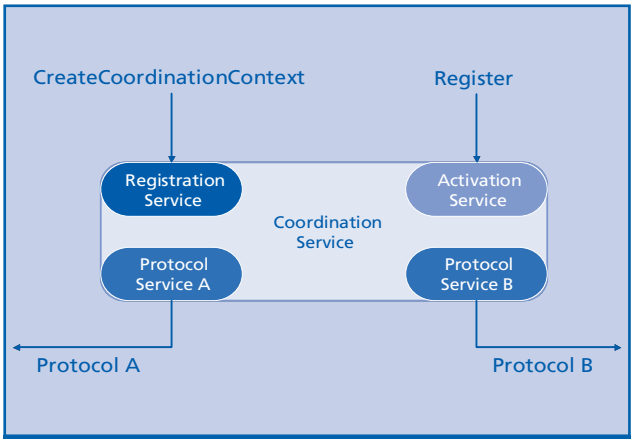


Figure 4.5: Service co-ordination

Figure 4.6: Atomic Transaction Protocol

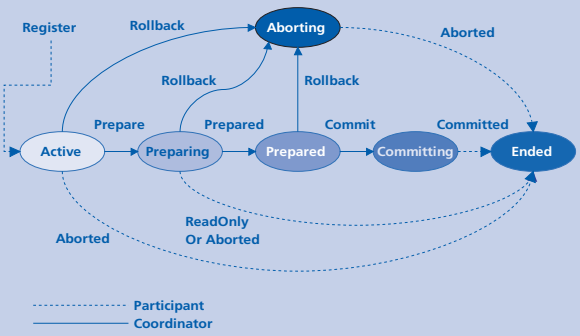
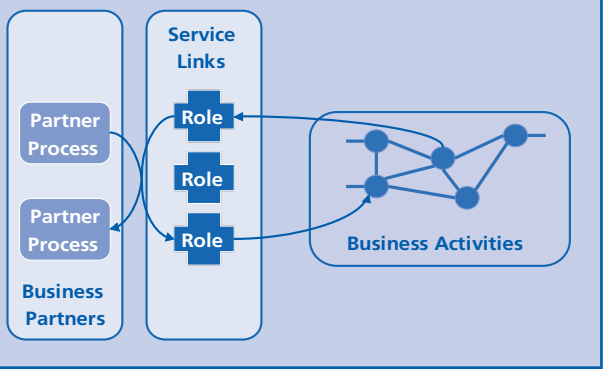


Figure 4.7: The service flow model



Web Services

Tom Welsh
Consultant

Management introduction

Two years after Microsoft's grand announcement of its new .NET strategy in July 2000, the IT industry has changed less than most might have expected. True, the media are full of information and comment about Web Services. But adoption has lagged, and even the necessary standards are not yet fully stable.

In this analysis, Tom Welsh looks at:

- ***the current state of Web Services***
- ***what you might expect to do when.***

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

The Web Services vision

There are many definitions of Web Services, from the poetic to the technical. But what is the essence of the Web Services concept? Simply this: extend the universality and ease of use of the Web to programs (including applications) as well as people.

When computer networks first appeared, each hardware manufacturer had its own style. For IBM it was SNA. Digital had DECnet, and so on. Where there was sufficient demand, gateways were provided between different network implementations but vendors did not encourage such 'leakage', for each aspired to make its own network the 'globally accepted standard'.

Then along came the Internet. Suddenly people could send email and transfer files anywhere in the world. Today, a computer that lacks TCP/IP, and thereby Internet connectivity, now seems somehow crippled.

This freedom is all very well, but it only applies to human beings. People sit at monitors and browse Web sites, exchange email or even experience music and video. But what about programs? After all, it is a poor kind of computer function if it cannot be automated. Just think of the possibilities if software on my computer could reach out halfway round the world and co-operate, or exchange data, with any one else's software.

This seems all the more sensible (and feasible) if you reflect that Web browsing or email are nothing more or less than remote program to program communication, with user interfaces added on as the first and last steps. Why not generalize the architecture, so that instead of receiving a page of HTML that can only be rendered on a screen, a client could accept all sorts of coded messages?

Is this real?

A standalone computer is limited in many ways. 'Programmatic' remote communication might alleviate some of its limitations. Three special cases immediately suggest themselves:

- **adding raw computing power**
- **providing access to data**
- **offering access to computational services.**

In the first, this is what well-known distributed applications like the SETI@home screen saver do already. Unable to pay for enough supercomputer time to analyze the output of its celestial scans, the Search for Extra-Terrestrial Intelligence (SETI) project came up with a screen saver program that uses a computer's otherwise idle processing time to do the necessary computation. Periodically, it connects to a central server to upload results and download fresh work units. Grid computing, one of the latest research directions, takes the same idea still further.

The second is just a classic client/server operation provided in a new guise — although the fact that the client connects initially to an HTTP server adds levels of flexibility. The server may provide the required data from a database running on the same machine, or through one or more application servers running in the background.

In the third, the client, conceptually speaking, asks a question. After a while, back comes an answer. This is different from the first: the client provides the software and the remote machine merely hosts it while it runs.

What advantages do these hypothetical modes of operation have over today's Web? It is mainly a matter of disintermediation — cutting out the middleman. It is well known that work flow software can sometimes slash the elapsed time to deliver a service from days or weeks to seconds, just by eliminating unnecessary delays. Straight through processing (STP) has the same objective of doing as much as possible at machine speed.

For instance, shipping companies provide Web and email based applications through which their customers can monitor the status of deliveries. With Web Services, if a storm or breakdown occurred at night or over a weekend, an automatic message could be sent to the customer's supply chain management application, which could take immediate compensating action. With today's human-based systems, nothing would even begin to happen until the employee responsible arrived in the morning (and perhaps had had the mandatory cup of coffee or tea).

Microsoft and IBM — and their visions

It was Microsoft that got the world excited about Web Services with its July 2000 launch of the .NET strategy. In some ways, .NET simply took current thinking about e-commerce that bit further. Consider, for instance, what Bill Gates said in his speech to Microsoft's Professional Developer Conference in October 2001: *"The power of the XML Web Services model is amazing. A company offering an online electronic-payment service can expose its service to partners, so that they can deliver it as part of their own offering — regardless of what platform they are using. An airline can link its online reservation system to that of a car-rental partner, so travelers can book a car at the same time they book a flight. An online auction company can notify bid-*

ders when they are outbid or have won an auction, or could partner with other firms to offer alternative shipping, fulfillment or payment options. XML Web Services help your business break free of its boundaries."

According to IBM, which decided to work closely with Microsoft around Web Services (though not necessarily in .NET form): *"Web Services are Internet-based modular applications that perform a specific business task and conform to a specific technical format. A Web service can be anything from a restaurant review service to a real-time travel advisory to an entire airline ticket reservation process. The modular technical format ensures these self-contained business services (from the same or different companies) will mix and match easily to create a complete business process. Businesses can dynamically publish, discover and aggregate a range of Web Services via the Internet; in this way, they can more easily and dynamically create innovative products, business processes, and value chains. Web Services can be delivered to any customer device (cell phone, PDA, computer, etc.) and can be created or transformed from existing applications."*

So much for the Web Services vision. It is time to consider how much has been delivered to date, and who is putting this innovative software to work.

What exactly are Web Services?

To find out just what Web Services really are, it is necessary to tackle the question from several angles:

- **first of all, how are they (Web Services) supposed to work?**
- **second, what are the standards — and how mature are they?**
- **third, what products are available that implement those standards?**
- **finally, but not least, who is putting the Web Services model into action — and with what results?**

Architecture

As we have seen, simplicity is the desired theme of the Web Services architecture. In theory, anyone should be able to create a Web Service — ideally with a few lines of code or, better still, by dragging and dropping. The result of creating this Web Service is that it should be immediately available to all authorized users. After all, a Web Service is just an XML message being sent over HTTP. Most of today's computers have the necessary protocol stack to handle this.

The three core standards are SOAP, WSDL and UDDI. The basic sequence (Figure 5.1) is this:

- **the Service Provider publishes a description of the services it has to offer, using WSDL and UDDI**
- **the Service Requester searches for (and, all being well, finds) a service that matches certain criteria**
- **the Service Requester binds to the service and invokes it, using SOAP.**

Any and all other specifications that come into play — such as WS-Coordination, WS-Transaction or WS-Security — are layered on top of these fundamental handshakes.

The standards

Much has been, and will be, written about 'Web Services standards'. But the fact is that, as of October 2002, not one single specification relating to Web Services has yet been adopted by a recognized standards body:

- **SOAP and WSDL are 'work in progress' at W3C**
- **UDDI and WS-Security have been submitted to OASIS, but have not yet been formalized.**

As for WS-Inspection, WS-Security, WS-Coordination, WS-Transaction and BPEL4WS, these have simply been proclaimed by IBM and Microsoft. It is said that this approach is much faster than the supposedly laborious proceedings of standards bodies. But it does have the drawback of not actually producing anything accepted as standard.

The Web Services Interoperability Organization (WS-I) was set up in February 2002 by Accenture, BEA, Fujitsu, HP, IBM, Intel, Microsoft, Oracle and SAP. Although such a collection of industry giants lent the new consortium authority, it explicitly denied any intention of creating standards.

Instead, its mission is to reconcile existing standards. It proposes to achieve this by publishing profiles that show which configurations are known to work together reliably.

SOAP

Simple Object Access Protocol (SOAP) is a lightweight protocol for the exchange of information in a decentralized environment — typically across the Internet or intranets. As such, it plays a role similar to that of IIOP (in CORBA and J2EE) or HTTP (in ordinary Web browsing). Based on XML, SOAP defines three parts:

- **an envelope that defines a framework for describing what is in a message and how to process it**
- **a set of encoding rules for expressing instances of application-defined data types**
- **a convention for representing remote procedure calls (RPCs) and responses.**

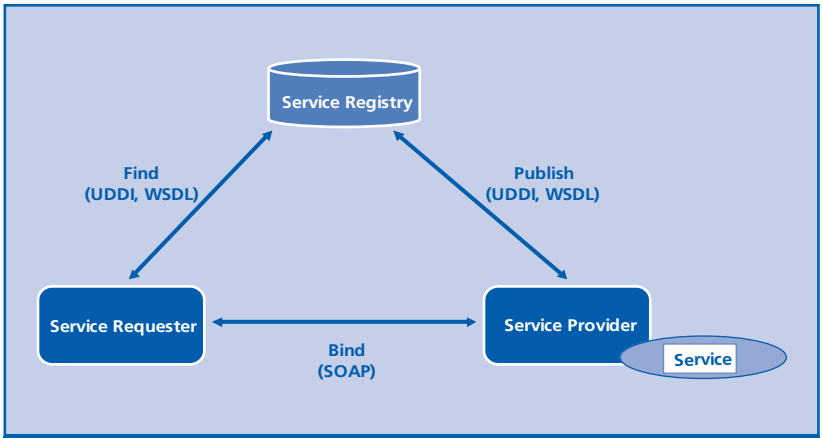


Figure 5.1: The three core standards

SOAP was submitted to the W3C in July 2000 by a group of companies which included IBM and Microsoft. The current version, SOAP 1.1, is used by virtually all products on the market. SOAP 1.2, a working draft of the W3C's XML Protocol (XP) working group, is a moving target. Among other improvements in version 1.2, the SOAP envelope and encoding schemas have been updated to be compliant with the XML Schema Recommendation.

This is not an unmixed blessing, as the XML Schema has turned out to be quite complicated and difficult. James Clark, who was technical lead for the W3C's XML 1.0 specification, has described XML Schema as "without doubt the hardest to understand specification that I have ever read".

UDDI

Universal Description, Discovery and Integration (UDDI) was announced in September 2000 by Ariba, IBM and Microsoft. Significantly, this was at the height of enthusiasm for e-commerce and online marketplaces, just before the 'new economy' bubble burst. This goes some way to explaining the new standard's relatively slow rate of uptake.

Based on SOAP and WSDL, UDDI allows providers to advertise the Web Services they are prepared to offer. Thus it helps to answer the question: how am I going to find what Web Services exist?

The equivalent of 'name servers' in the CORBA and Java worlds, UDDI operates on three separate levels:

- **white pages (address information)**
- **yellow pages (business categories)**
- **green pages (technical details of how transactions are to be conducted).**

One interesting future direction for UDDI is integration with the Lightweight Directory Access Protocol (LDAP). BEA, Novell and Sun have all expressed the intention of moving in this direction, which would at least prevent UDDI being passed over in favor of the better-known and more mature LDAP.

WSDL

The Web Services Description Language (WSDL) is an XML-based contract language jointly announced by Microsoft and IBM in September 2000. It has been called 'the XML equivalent of a resume' — telling:

- **what a Web Service can do**
- **where it lives**
- **how to invoke it.**

Its role is similar to that of Interface Definition Language (IDL) in CORBA and COM+. (Unfortunately, however, Web Services developers usually prefer to start with code and generate WSDL from that — an irrational and inefficient approach.)

W3C announced the working draft of WSDL 1.2 in July 2002. It tidies up the earlier specification, removing redundant and non-interoperable features and providing for bindings to SOAP 1.2, HTTP and Multipurpose Internet Mail Extensions (MIME).

WSDL 1.2 is also compatible with the XML Schema and the XML Information Set. However, as in the case of SOAP 1.2, vendors are reluctant to implement WSDL 1.2 until it is finalized and frozen.

WS-Inspection

WS-Inspection is the result of yet another co-operative effort between IBM and Microsoft. It is an XML format that

can be used to find out what services are available on a specific Web site, and how these can be invoked. This approach is complementary to UDDI, which looks up services in a central directory, not worrying about where they are served from until later, and WSDL, which gives details about how to invoke a specific service.

WS-Security

IBM, Microsoft and Verisign announced WS-Security in April 2002, together with a roadmap document entitled 'Security in a Web Services World'. This roadmap — which depicts the future development of Web Services security — was authored by IBM and Microsoft, with no mention of the third author (and security specialist) Verisign.

WS-Security — which defines a standard set of SOAP extensions and mechanisms for the exchange of secure, signed messages — is largely a framework into which other security specifications can be slotted. It is compatible with state-of-the-art security schemes like:

- **Kerberos**
- **public key infrastructures (PKI)**
- **current W3C/IETF work including XML Signature and XML Encryption.**

At the same time, six subsequent specifications were named. They were:

- **WS-Policy**
- **WS-Trust**
- **WS-Privacy**
- **WS-Secure Conversation**
- **WS-Federation**
- **WS-Authorization.**

In July 2002 OASIS agreed to host the further development of WS-Security, setting up a new technical committee for the purpose — with co-chairs from IBM and Microsoft. Presumably some of the other prominent security standards — such as Security Assertion Markup Language (SAML) and XML Key Management Specification (XKMS) — will be accommodated within WS-Security over time.

WS-Coordination

The latest set of de-facto Web Services standards was announced in August 2002 by BEA, IBM and Microsoft. WS-Coordination, WS-Transaction and BPEL4WS were launched together and, while the last-named stands apart, WS-Coordination and WS-Transaction are often used together.

WS-Coordination provides an extensible framework for protocols used to coordinate the actions of distributed applications. This includes those that need to reach consistent agreement on the outcome of distributed transactions.

WS-Transaction is designed to be used in conjunction with other parts of the GXA stack, such as WS-Coordination and WS-Security. WS-Transaction tackles the hard problem of long-lasting Web transactions. In traditional database applications, no transaction lasts longer than a few seconds at most. Web transactions may last much longer. They do not always need to respect the traditional ACID properties. WS-Transaction is for those that do, while WS-Coordination supports those that do not.

BPEL4WS

Business Process Execution Language for Web Services (BPEL4WS) is generally acknowledged to be one of the worst acronyms ever seen or coined. In the by now familiar way, it combines a Microsoft language with an IBM one — in this case:

- **BizTalk's XLANG**
- **the Web Services Flow Language (WSFL).**

The result is more comprehensive than comprehensible.

Available products

When looking at where available products exist it is relatively straightforward to assign the mass of Web Service enabled products to one of three main groups:

- **development toolkits**
- **application servers**
- **applications.**

The best-known toolkits are those from Microsoft and IBM, the leading proponents of Web Services as well as the two largest software vendors in the world. Microsoft's Visual Studio.NET is generally considered to be a fine integrated development environment (IDE). As well as allowing Web Services to be created and consumed with a line of code, efforts have been made to keep it compatible with its predecessor Visual Studio 6. Microsoft is doing all it can to encourage the use of its new language Visual C# ('C Sharp'), although Visual Basic's popularity will die hard.

IBM has invested heavily in writing Web Services tools based on Java — just about the only language not supported by Visual Studio.Net. Its WebSphere Studio Applica-

tion Developer (WSAD) provides everything necessary to create and consume Web Services, starting from a J2EE application server such as WebSphere. Another differentiator from Microsoft is that IBM is not obsessed with the language dimension, which it regards as misguided. IBM prefers to focus on creating something that works rather than on the programming model.

IBM has donated a number of production-quality tools to the Apache Software Foundation, whose Axis (also known as Apache SOAP 3.0) is an open source toolkit that is compatible with both Microsoft-style and Java-style Web Services.

Other leading toolkit suppliers include BEA, Cape Clear, The Mind Electric and Systinet. But there are literally dozens of offerings from which to choose, both proprietary and open source.

Turning to the second category (above), virtually all application server vendors have hastened to add Web Services capability to their products. This is not necessarily a particularly demanding technical task now that the standards are gradually stabilizing. It has, however, become absolutely necessary as a Web Services capability is becoming a 'must have' requirement for many purchasers.

BEA, IBM, Iona, Oracle, Sun and, of course, Microsoft, have all added current Web Services support to their respective application servers. Dublin-based Cape Clear has pioneered the technical challenge of front-ending a J2EE application server with a complete set of Web Services features. Silverstream, which was recently acquired by Novell, brought a strong Web Services emphasis with it which — together with Novell's great strength in directories — may bring the latter back into the limelight.

Turning, finally, to applications, it is worth noting that this is what all the fuss is really about. Toolkits and platforms are nothing more than a means to an end — and that end is offering specific services (usually for a consideration) over the Internet or, in the early days, corporate intranets.

Naturally, all the prominent e-commerce specialists of the late 1990s are crowding in on what is widely perceived as a new technology with the potential to resurrect their visions. Ariba and Bowstreet, partnered prominently with IBM and Microsoft, are working out the first Web Services specifications, and Commerce One has also been involved.

However, the vendors that stand to gain (or lose) most from the arrival of commercially viable Web Services are those specializing in Enterprise Commerce Management (ECM), an umbrella term covering:

- **Enterprise Resource Planning (ERP)**
- **Customer Relationship Management (CRM)**
- **Supply Chain Management (SCM)**
- **Product Line Management (PLM)**
- **Enterprise Asset Management (EAM).**

SAP, PeopleSoft, JD Edwards, Siebel and many of other competitors have announced plans to make some or all of their application suites available in the form of Web Services. If, and when, this actually happens, it may be good news for customers. They, at last, will be able to pick and choose the service components that are most appropriate to their needs.

This section started with Microsoft, and it will close with it too. BizTalk Server and some of Microsoft's other application offerings make extensive use of Web Services — although it is worth noticing that they can also provide alternatives, such as Microsoft Message Queuing, as well.

Management conclusion

The Web Services vision holds out a lot of promise. But there are several serious issues that must be dealt with before that promise can be realized. In approximate order of priority, these include:

- ***agreeing on a set of vendor-neutral, legally unencumbered standards that can be efficiently implemented***
- ***guaranteeing security, integrity and privacy***
- ***attending to performance and quality of service***
- ***agreeing suitable shared vocabularies***
- ***making sure that, once extensive sets of automated Web Services are built, they do not have negative repercussions like those caused by automated securities trading.***

Meanwhile, so great are the alleged opportunities that many companies are not waiting. Instead, they have launched a wide variety of products for everything from building Web Services to procurement, research and notification. But success stories are limited right now. Yet, with so many people working on the problems of Web Services, progress should be rapid. That is not to say that all the problems will be solved right away, but at least we should soon have a clearer picture of what is, and is not, feasible in the short term.

Security — rebuilding confidence in Web Services

Mark Lillycrop
Principal
Arcati

Management introduction

"We really haven't done everything we could to protect our customers. Our products just aren't engineered for security." This sheepish admission by Microsoft's Senior VP for Windows Development, Brian Valentine, at a recent .NET developer conference must have surprised many in his audience. Let's face it: a frank mea culpa from a leading industry vendor is a rare occurrence. But Mr Valentine was simply underlining one of the fundamental obstacles to progress in Web Services — the lack of basic, reliable security in the areas that matter most.

Web Services will live or die on the computer industry's ability to convince the market place that they are as secure and reliable as the IT architectures that they replace or complement. Over the coming months, user organizations need to see the emergence of consistent middleware-based security standards — in the areas of:

- ***authentication***
- ***encryption***
- ***non-repudiation***
- ***threat management***

— to support their new breed of applications. In this analysis, Mark Lillycrop examines what has been happening and what might yet occur.

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

Building confidence

Security has risen rapidly to the top of the agenda in IT departments, as the danger of real and perceived threats to the integrity of corporate and private data reaches epidemic proportions. Figures from Internet security watchdog CERT, based at Carnegie Mellon University, indicate the extent of the problem:

- **in 1997, it logged 2134 security incidents and 311 areas of vulnerability**
- **by 2001, these figures had risen to 52658 and 2437 respectively**
- **they look set to double again this year.**

Every one of these cases represents a risk that businesses and/or government departments will:

- **face denial of service attacks**
- **experience theft of critical or confidential information**
- **be plagued by viruses, corrupt data, or worse.**

CERT is not the only body expressing concern. Even the FBI has got in on the act. Its recent survey, published in conjunction with the Computer Security Institute, revealed that nearly 40% of large organizations had reported computer crime to the police, up from 17% five years ago. But the number of companies affected is likely to be considerably higher, as many decide not to report attacks in case it damages customer confidence.

And confidence is what it is all about. Using a Web-based delivery mechanism for business applications of any sort involves a high level of trust on the part of both the end customer and the IT professionals managing and monitoring such services.

Early in 2002, Microsoft responded to the mounting criticism levelled at security weaknesses in its products by announcing the Trustworthy Computing initiative (Figure 6.1). In an internal memo at the time, Bill Gates pointed out that, despite the company's commercial success at software development, "all those great features won't matter unless customers trust our software. So now, when we face a choice between adding features and resolving security issues, we need to choose security."

Fine words, even if they do have a somewhat hollow ring to them. Gates is clearly indicating that security management (not to mention availability, privacy and integrity — which are also targets of this new program) will reside much closer to the hearts of his developers in the future. If this strategy is successful — and future versions of Windows XP, .NET and other mainstays of the Web Services world can exhibit business-class security characteristics — it will certainly address part of the issue.

Brian Valentine's admission above was not so much an expression of helplessness as an assertion that his company is now changing tack and facing up to its responsibilities. To this end, Microsoft has:

- **introduced mandatory security training for every Windows engineer**
- **assigned 'ownership' (and hence accountability) to every line of code**
- **incorporated security formally into the product development process**
- **improved the tools provided for customers to test for weaknesses in their product configurations.**

But it is a tough call. Trustworthy Computing is starting from a position of extreme distrust.

Even as this is written, Microsoft has announced three serious flaws affecting its Virtual Machine, one of which "could enable an attacker to gain complete control over a user's system." The flaws affect all versions of Windows, and the only offered course of action is for all Windows users to implement a critical patch.

That the warning is made in good faith is not disputed. But the logistical nightmare of 'patching' the entire Windows community every time a flaw is discovered — a point made in the August, 2002 FINANCIAL MIDDLEWARESPECTRA by Don Haile, the CTO at Fidelity — is hardly conducive to extending the reliance of the business world on Web Services.

Legacy systems and interoperability

The very fact that Microsoft is taking its software vulnerabilities seriously is an important step towards building confidence. But other important questions remain:

- **what about the huge base of legacy products, on which much of the existing e-business infrastructure depends?**
- **what about interoperability with other suppliers' products?**

On the first of these issues, the industry may well be forced simply to draw a line under the weaknesses in many older products, and move forward. Vulnerabilities in earlier releases of Windows, and design flaws in products such as

ActiveX, cannot easily be rectified, and business-critical Web Services will rely on newer tools and standards.

This, of course, places additional pressure on corporate users to upgrade to the latest products. But, in so many enterprise networks, it is often not possible to locate every last legacy application.

Interoperability is an equally thorny issue. Authentication and intrusion detection policies determined by Microsoft are not necessarily applicable or relevant to other suppliers. But Web building-blocks — such as XML and SOAP — are increasingly defining other vendors' strategies. A consistent approach to security within middleware is absolutely critical here.

While Microsoft is one of the biggest fish in the e-business pond, and can be blamed for patchy standards in the Web Services arena in the past, one thing is clear. Web Services need an industry-wide approach to security management if they are to win sufficient public confidence to enable organizations, and individuals, to adopt them.

Delivering security for Web Services

Security services for tools such as XML and SOAP are being provided in two ways. One source of expertise is a small but very influential group of companies that offers products for the Web Services market. This group is gradually extending its range of tools in order to support emerging standards.

Of these, the best known and most innovative is probably Check Point, which develops firewall and VPN products. The company's recently announced Feature Pack 3, for example, allows firewalls to be configured to check the structure of XML documents and SOAP messages, an area which is of increasing concern to organizations passing large numbers of complex documents over the Web. This vendor has also been showing off its Application Intelligence technology for SOAP and XML, geared to providing secure information sharing between Internet applications.

The second source of security expertise for Web services lies in industry consortia, of which two are particularly worth watching at the moment:

- **WS-Security**
- **the Liberty Alliance Project.**

Such industry bodies are notorious for generating vast amounts of hot air while achieving very little. But the whole nature of Web Services, and most vendors' desperation to generate some enthusiasm for them at a time of industry-wide recession, suggest that these bodies have a realistic of reaching consensus rather faster than normal.

WS-Security

WS-Security was originally developed jointly by IBM, security specialist VeriSign and Microsoft. It has recently been submitted to the Organization for the Advancement of Structured Information Standards (OASIS) which has the unenviable task of managing its future development.

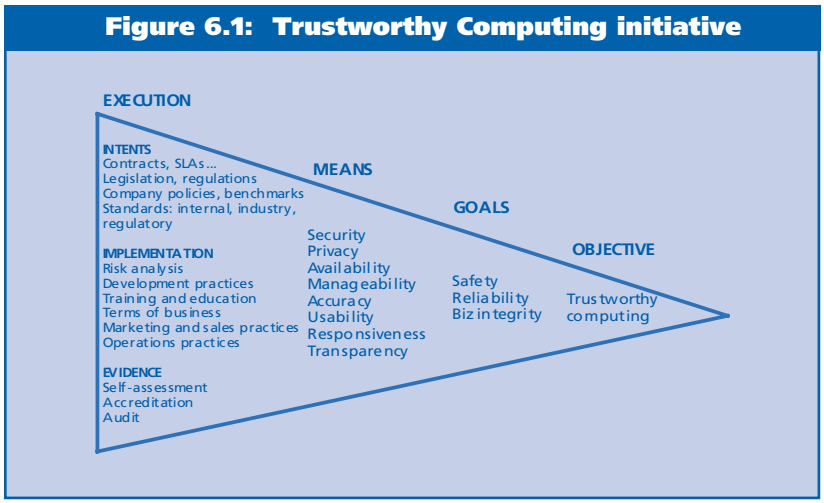
One of OASIS' first steps was to form a technical committee to build on the specification. The organization has brought together nearly 60 companies — who all want a say in the way that WS-Security evolves

WS-Security is essentially an extension to the Microsoft SOAP standard, providing what its originators describe as "quality of protection through message integrity, message confidentiality and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies."

The specification also:

- **provides a general-purpose mechanism for associating security tokens with messages**
- **supports a range of token formats**
- **describes how to encode binary security tokens, including X.509 certificates and Kerberos tickets.**

Figure 6.1: Trustworthy Computing initiative



It is still early days for WS-Security. A number of omissions have already been identified. The technical committee has still to push the working draft forward as quickly as possible. But the forum will provide an essential test-bed for interoperability of Web Services in coming months.

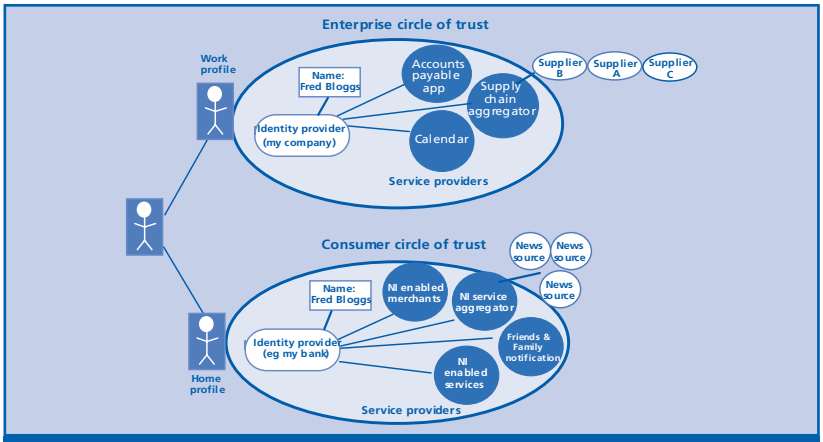


Figure 6.2: Federated network identity

The Liberty Alliance

Founded in October last year, the Liberty Alliance Project is in many ways a complementary movement to WS-Security, although (partly because of its origins) it is often seen as being competitive. With an impressive management board line-up — including such companies as American Express, Vodafone, General Motors and United Airlines — the real impetus for the body comes from the likes of Sun Microsystems and HP (arguably in response to IBM, Microsoft and Verisign).

The Liberty Alliance focuses strictly on what it calls 'federated network identity'. The immediate goal is to provide users with a single online identity and personal profile, access to which would be determined by the user him or herself (Figure 6.2).

The ultimate aim is to build an extensive network identity account for each person, including:

- **entertainment preferences**
- **educational history**
- **financial information**
- **credit card details**
- **passwords.**

But the first step is to achieve a single, multi-vendor Web-based sign-on — a very attractive objective. It is one, however, that has eluded most security specialists so far.

Liberty Alliance is working in an area that is both politically sensitive and commercially explosive, challenging the Microsoft Passport approach to network identity. While unauthorized access to personal data is acknowledged as a serious problem, carefully managed authorized access can remove many of the frustrations of using Web-based Services today.

The Liberty Alliance ranks are swelling as fast as those of WS-Security. The reason is simple. More and more vendors see the need to be part of interoperability projects that are becoming so critical to future Web application development.

User responsibilities

WS-Security and the Liberty Alliance reflect a healthy change in the way IT service and product suppliers are viewing security and the need to work together. Some responsibility, though, inevitably has to rest with users, and with the vendor/user relationship.

CERT, whose statistics were cited earlier, suggests that 95% of recorded attacks could have been avoided by applying known fixes to vulnerabilities. This is a damning indictment:

- **on the level of post-sales support provided by some vendors**
- **on the understanding of security requirements displayed by many users.**

Some hackers and viruses require enormous ingenuity to combat. But most hacking cases can be put down to copycat mischief-makers seeking out sloppy network configurations and weak default settings. The majority of products are easy enough to equip with basic protection, if those who support them are aware that they need to change defaults and simply close the doors that are usually left open on delivery.

Inevitably, businesses vary greatly in the rigor of their security strategies and in the priorities that they set. There is a limit to the amount of influence that governments can exert here, although some pundits would like to see a much tougher line taken, if only so that we could all place more confidence in Web Services in future.

The recently announced US government draft recommendations for 'A National Strategy To Secure Cyberspace' were remarkable in their restraint and lack of regulation. Instead the focus was on the responsibility of companies

and individuals to be thorough and vigilant in their attempts to make sure that critical systems remain unaffected by cybercrime.

Likewise, the OECD's recently update security guidelines, jointly devised by the body's 30 member states, are generalized in their nature. While espousing simple common sense, they are also open to widespread interpretation — particularly the resolution that says: "Security of information systems should be made to be compatible with the essential values of a democratic society."

Where now?

So without firm rulings from government (which, in any case, would be resisted by many enterprises), where do we go from here in building a foundation for Web Service security? Three aspects need to improve, quite apart from the development of new standards that has already been discussed.

First, user companies need a detailed security policy that is implemented consistently from the Board down. The policy needs to:

- **include the management of data on laptops and handheld devices, and the way that information is moved around the organization and between co-operating partner companies**
- **address the way access authorization is applied to complex electronic documents, and the way that privacy requirements are observed**
- **look at every aspect of multiple-level authentication, non-repudiation, encryption and threat management — in detail.**

This presents a formidable challenge for most organizations, where security policies have evolved from existing IT guidelines and are poorly understood at senior management level.

This moves us onto the second issue: education. CEOs and senior executives, including most especially CIOs, need to understand the risks they face, and the best ways to combat them. Some security vendors, for example, run ethical hacking workshops for their customers, providing them with a clearer perspective of the cybercriminal mind-set.

CIOs also need to communicate regularly with business partners and government colleagues to exchange information on areas of risk and potential threats. Large corporate buyers have traditionally been reluctant to share security information in case it leaves them more exposed. In the Internet environment nobody can afford to be ignorant of the dangers they face.

Thirdly, we all need to improve the flow of security information between vendors and users on the availability of patches and fixes and the presence of potential security flaws in products. In many cases, customers know they can obtain this information from suppliers, but are unsure how to go about it.

This is not a one-way street. Many vulnerabilities are picked up by users themselves, but the means for feeding reports back to the supplier are often unclear and sometimes the vendor is reluctant to accept the feedback at all.

Management conclusion

Mature security middleware is an essential precursor to the long-term acceptance of Web Services. Vendors need to work together closely to achieve this.

But, in today's software environment, the real challenge is for vendors and users to work together, to develop an efficient communication channel and to use their complementary strengths to manage the potential risks to corporate information systems. Moreover, both organizations and their managements need to understand the limitations of each component solution and be particularly aware of the gaps between the individual standards and tools.

As the introduction to the WS-Security specification puts it: "WS-Security is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models and encryption technologies. Implementing WS-Security does not mean that an application cannot be attacked or that the security cannot be compromised."

The real danger comes from over-estimating what can be achieved with security standards in a highly complex and constantly changing environment. In the end, nothing can replace vigilance and constant re-assessment of corporate security policies.

XML-Aware networking: a framework for XML security

Eugene Kuznetsov
President
DataPower Technology

Management introduction

Technically, XML is a way of marking up structured data. Because of its simple syntax, Unicode support and extensibility, it is gaining widespread use as a universal data interchange format. Some efforts — notably Web Services — are going further, specifying an entire distributed computing infrastructure built around the exchange of XML-defined messages.

The adoption of XML is associated with the broader move away from secure private networks to using the Internet for e-business communication. Like many new technologies that enhance the exchange of information, XML, if not addressed properly, can make it easier for unauthorized parties to access sensitive information. Therefore, mission critical XML applications create a whole new set of security concerns.

In this analysis, Eugene Kuznetsov of Datapower:

- ***examines XML-aware networking***
- ***describes a framework for XML security***
- ***examines the seven key XML-specific security issues that must be addressed going forward.***

All rights reserved; reproduction prohibited without prior written permission of the Publisher.
© 2002 Spectrum Reports Limited

Looking back

Taking a look at history, the rapid growth of the Web in its early days quickly meant that security, performance and reliability needs escalated well past those offered by a single server. To take one case in point, server load balancing (SLB) provided the ability to have a group of servers appear as a single Web server to users.

This virtualization of Web servers was initially done by special software deployed on the Web servers themselves but very quickly SLB functionality moved from software into content-aware network hardware, both appliances and switch grade products, delivered by companies such as:

- **F5 Networks**
- **ArrowPoint Communications**
- **Alteon WebSystems.**

Hardware load balancers provided the performance and reliability demanded by the ever-expanding Web server farms running the Internet's largest sites. In a typical installation, incoming page requests were load-balanced among dozens of individual Web servers. In consequence, the load balancing algorithms themselves quickly became more complex: URL switching, cookie redirection and SSL ID tracking are examples of how complexity increased. In addition, Web application developers became aware that the infrastructure's capability influenced application design and could make the use of an SLB possible or more beneficial.

Today

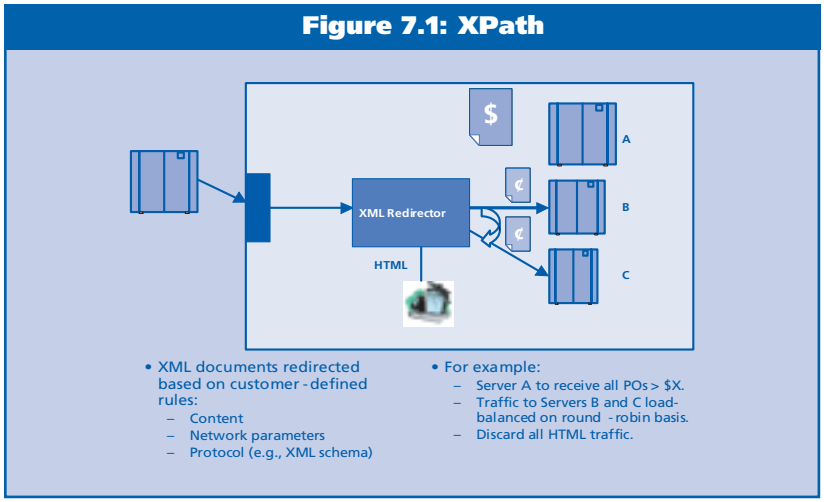
Today the security implications of Web services have prompted heated debate both in the XML and the security communities. For example, security expert Bruce Schneier, CTO of Counterpane Internet Security, criticized the Microsoft implementation of SOAP in a Crypt-Gram Newsletter: "Implementation of Microsoft SOAP, a protocol running over HTTP precisely so it could bypass firewalls, should be withdrawn. According to the Microsoft documentation: 'Since SOAP relies on HTTP as the transport mechanism, and most firewalls allow HTTP to pass through, you'll have no problem invoking SOAP endpoints from either side of a firewall.' It is exactly this feature-above-security mindset that needs to go."

SSL, content switching and content caching grew up with the Web, evolving their features in response to new applications and protocols, and enabling the tremendous growth of the Web. Although few think of this fact today, a heavily trafficked Web site cannot stay up for more than a few seconds without the content distribution network and the load balancers which enable it. Indeed, it was the HTTP-aware network infrastructure that helped overcome the early Web's performance, reliability and security problems — and enabled it to be used for the kinds of applications (for example, large news portals or online stores) which fueled its growth even further.

XML awareness

However, while SOAP and XML-RPC are built on top of HTTP — and can therefore flow through these old load balancers and accelerators — the content-aware network equipment designed for Web traffic does not know anything about XML-encoded data or the new protocols accompanying it. In order to be really useful for next-generation protocols, therefore, network devices must become 'XML-aware', or capable of looking further up the network stack — past TCP, SSL, HTTP and parsing the XML messages themselves.

Figure 7.1: XPath



But, just as performance issues are cause for proper planning rather than abandonment of XML or Web Services, so are security concerns. They must be addressed methodically from a project's start and they are certainly not a good reason to give up on the benefits of XML technology.

At the same time, computer and network security has evolved into a very large and varied vendor sector. One of the reasons that securing XML Web service applications is so difficult is that it involves several different security issues, each of which must be separated and addressed individually. Examining some XML and XML secu-

rity elements, in this context, should be instructive.

XML Redirectors

XML Redirectors offer basic routing functionality for XML (they are also sometimes referred to as 'XML switches' or 'XML routers'). Because they are the simplest kind of XML-aware 'box', they were the first ones to become commercially available (Intel offered the now discontinued NetStructure XML Redirector over two years ago).

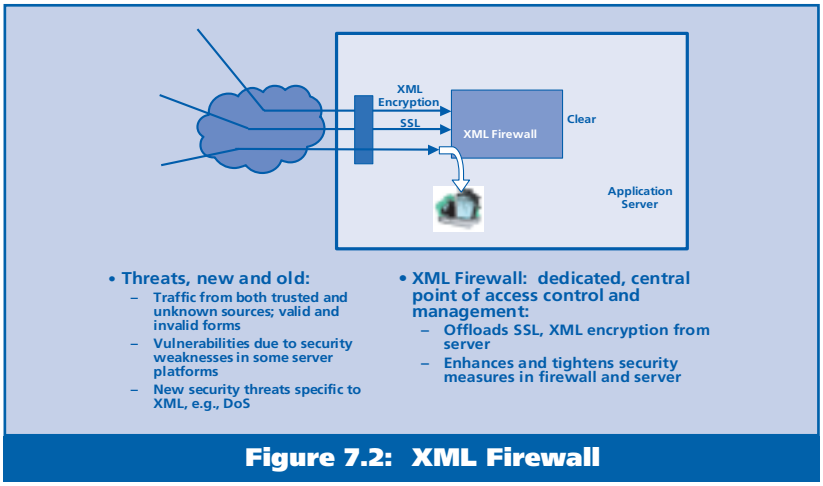


Figure 7.2: XML Firewall

Such a device is placed between servers and incoming XML traffic. It redirects XML messages between different servers based on content. For example, all purchase orders over $50,000 might be redirected to a fast application server, while all (lower value) orders might be sent to a slower server. In addition to this semi-trivial QoS (quality of service), the same device might send messages in a particular XML vocabulary to the server most capable of processing them. Alternatively, it might separate XML-RPC and SOAP messages. The routing rules are specified using either proprietary pattern-matching languages or a limited subset of XPath. (Figure 7.1)

XML Accelerators

XML Accelerators are network devices which aim to speed up either the flow or processing of XML messages. They are desirable because XML processing performance can be a barrier to adoption of XML technologies for business-critical applications. An XML Accelerator has the ability to offload one or more resource-intensive XML processing functions onto purpose-built hardware.

For example, bottlenecks associated with XML parsing and XSLT are a frequent and well-known problem. XML Accelerators can perform XSLT transformations outside the application, thereby greatly improving transaction throughput and reducing page load times. Utilizing standards — such as 'xml-stylesheet' processing instructions — can make it easy to drop in an Accelerator into an existing environment.

Other types of XML processing can also be performed 'in the network', with an XML Accelerator taking over the bottlenecked functionality from the application server. Of course, in order to take over processing, the Accelerator cannot implement just a subset of a specification; it must be fully conformant to the standard.

To be considered an 'Accelerator', a device must offer much better performance than a general purpose server. In fact, the best Accelerators will be quite unlike general purpose servers. They will:

- **consist of custom-built gear**
- **lack hard disks, keyboard or screens**
- **be controlled using existing network management systems and interfaces.**

This is similar to the SSL accelerators or Web caches that have been speeding up websites and e-commerce applications for years by providing a single-purpose, offload function. Since SOAP and other Web Services protocols are all XML-based, their throughput is intrinsically dependent on XML performance.

Expect the XML Accelerator space to become quite crowded as a number of vendors offer highly sophisticated devices delivering:

- **better XSLT performance**
- **high-speed schema validation**
- **XML parsing**
- **related or other functions.**

XML Firewalls

XML Firewalls (Figure 7.2) are true to their name. They play the same role for XML traffic that traditional IP firewalls do for the rest of the enterprise network.

Security is rapidly becoming a hot topic in the context of XML and Web Services, with good reason. If there is one issue that can stop (or rollback) the advance of XML messaging systems, it is security (or its failings).

An XML (or SOAP) firewall intercepts incoming and outgoing XML messages and delivers one or more of the following security functions:

- **assessing how well the XML is formed**
- **checking (and screening out) of invalid XML**
- **providing XML denial-of-service attack protection**
- **validating against standards (for example SOAP) or custom user-specified XML schemas or DTDs**
- **verifying XML signatures**
- **encrypting sensitive message fields, as necessary.**

All of these functions require XML parsing and transformation, functions that traditional firewalls cannot provide. Because they are CPU-intensive, this requires purpose-built hardware. Furthermore, it would only take one well-publicized security fiasco in a Web Services application at a Fortune 500 company to stall Web Services initiatives in the other 499.

An XML Firewall is on the wish list of many network administrators because it:

- **picks up where the existing firewall leaves off**
- **offers a measure of XML security and access control at the network level.**

Without it, an administrator's only options are either to attempt to block all XML network traffic or rely on the growing number of XML-enabled applications to be the sole layer of security. As a result, even today, deployment of externally-facing Web Services applications are often held back by the concerns of the corporate network security group about opening up port 80 to incoming SOAP traffic or just the very notion of offering external RPC access to the back end systems in an enterprise.

Technical challenges

Designing and building a truly useful XML-aware network device is an extremely difficult technical undertaking. Because enterprise network equipment is expected to function at wire speed — 100 megabits per second — the same is required of the XML processing core embedded in any XML device.

Smart decisions also need to be made about what kind of XML functionality should be put into the network without introducing additional complexity into the overall architecture. For example, manual setup of transformation rules can be cumbersome if not designed carefully. The complexity of configuration can be greatly eased by offering integration with XML development tools.

A good XML network device should integrate seamlessly with application software, preserving XML standards compliance while delivering the kind of performance expected of network devices.

In this context, it is important to note that the network infrastructure is composed of both:

- **network equipment**
- **network services.**

XML network equipment

Although XML-aware network equipment is much further along than network services, we can look forward to some changes in how complex applications are built. Looking again to the HTTP-aware network, during the Web's early growth it became clear that performance and reliability needs of the largest Web sites were well past those offered by one server cluster.

Flash crowds — spurred by major events like the Superbowl or the Olympics — can overwhelm any single network or server group. The Internet's growing global reach has meant even longer waits and growing bandwidth costs.

XML network services

Distributed edge caches have moved content closer to users — reducing latency, load on origin servers and ISP bandwidth costs all at once. The Content Distribution Network (CDN) is a network service employing thousands of caching servers located at ISPs across the globe and first offered by startups Akamai, Digital Island and several others. Often CDN's are combined with global load balancing to direct a user's requests to the closest server pool or one which can be reached via the cheapest or least-congested network link.

Similar kinds of 'overlay' networks are now being created to offer guaranteed delivery of business documents or transactions through the Internet. Such a network service is designed to allow a customer to send a purchase order 'into the network cloud' and be assured that it will be delivered to its destination despite any failed network links, security challenges or other disruptions. Queuing servers and encryption are used to offer a sophisticated system. Asynchronous Web Services specifications and HTTP-R are

technologies that are finding application in this kind of network service.

Online machine-readable directories for XML Web Services are another obvious network service. It is no accident that some of the biggest existing network service providers are involved in UDDI and similar initiatives. Their hope is that UDDI — or a similar directory initiative — can become a next-generation domain name system of sorts for Web Service end points.

Of course, just because a resource is publicly listed, it does not mean that it should be accessible to all or without charge. In the security and payment area, innovative startups are creating automated access brokering services based on signed cryptographic assertions — often using SAML — which will enable both pay-per-use XML Web Services and new business models for existing content providers.

All of these are merely examples of the kinds of XML-aware network services that XML developers will likely use in a year from now. With an understanding of these XML elements, that can be used toward securing XML applications, let us look at the seven key XML-specific security issues that must be addressed going forward.

Secure transport

For all e-business applications it is essential to protect the network link against eavesdropping and unauthorized access. This task is best addressed using well-established transport security protocols, such as IPSEC (network-level) or SSL/TLS (connection-level). This will:

- **ensure that all network nodes attempting to connect to a server are authorized to access it**
- **help prevent people from intercepting or tampering with messages as they travel across the network.**

SSL/TLS (commonly referred to as just 'SSL') has three frequent uses. It can be used to:

- **encrypt a TCP connection between two nodes (the client and the server, for SSL purposes), thereby protecting it against snooping and forgery**
- **authenticate the server to the client, whereby the client can be certain that it is communicating with the desired server**
- **authenticate the client to the server, offering access control.**

Most XML networks today use SSL because it is easier to deploy due to the existing SSL infrastructure and expertise that can be re-used from past e-commerce Web site projects. It is important to note that, depending on whether and how certificates are used, SSL may provide as much as full secure access control or as little as snooping-protection.

XML filtering and fine grained access control

Although transport-level security is an essential first step, there also exists a need for controlling access at a per-message or per-application basis. The data and applications on the back end servers are valuable and extremely sensitive. Therefore, even though the network connection may offer authentication, it often makes sense to limit certain operations by type, for example by requesting application, and other rules which may be part of an overall XML security policy. This is similar to the IP firewalls widely deployed today, which examine all incoming and outgoing packets for validity and apply user-defined filters. The main difference is that the traditional firewall rule may be something like: 'no host can connect from the outside to the file server', while XML filtering rule may be: 'do not allow any XML message with an '<action type='delete'>' element through to the archiving server.

Once the security of the back end server can no longer be guaranteed by simply shutting it off from the world, higher-level packet filtering and application-aware security solutions will be required. SAML and WS-Security are two of the new specifications for enabling fine-grained access control for XML messages.

A whole new breed of XML-aware firewalls and specialized VPNs will use knowledge of XML protocols and security policies to enable secure connectivity. These devices will also have to defend against a new breed of DoS attacks, support XML-aware QoS and integrate with much of the existing network security infrastructure.

XML and Denial of Service (DoS) attacks

As more network applications use XML as the means to communicate, there is an increased probability of DoS attacks that exploit the very flexibility of XML that makes it so popular. A 'typical' DoS attack transmits invalid packets to the target host or router, forcing it to expend resources to service the phony requests and thereby degrade or deny service to legitimate requests. The well-known TCP SYN Flood and Unix Process Table Overflow attacks exploit the ability to initiate a connection request without completing

it. Other attacks send malformed packets designed to induce the network node to consume processing or bandwidth resources.

In all cases the ability of a server to stand up to such an attack is based on its ability to reject quickly an invalid packet or connection request. As parsing and validating XML are resource-intensive, it may take much longer to determine that a particular XML-RPC encoded item is 'corrupt' than to do so for a binary RPC or TCP packet.

Therefore, the ability to determine quickly that a particular XML-encoded request is invalid and reject it without further processing or consumption of resources will be absolutely essential to the survivability of any XML-aware network.

Integration with existing security infrastructure

The extensive development required to meet the security requirements of XML networking does not mean that the existing security infrastructure will be replaced. Rather, the new XML-aware security products and processes must integrate with systems that are already deployed — from firewalls and TCP DoS filters to managed VPN installations and complete PKI systems. For example, OSI layer 6 protocol processing, back-end server load balancing and TCP termination final layer of security may be handled by XML-aware network equipment, while the payloads may travel through a previously deployed VPN.

This makes sense technologically, because of the complementary functions of a simple TCP firewall used in IP-VPNs and a complex XML protocol firewall that is part of XML aware network equipment. But it also protects both the financial investments and the customer trust in the existing security infrastructure.

This further strengthens requirements for XML network equipment to be able to communicate with, and co-operate with, other network devices and nodes to implement security, QoS (Quality of Service)and other policies.

Schema validation

The logical structure of an XML message is defined using an XML schema file. Validation of an XML message — to ensure that it conforms to the expected schema — is an essential part of security processing.

By performing validation on all XML documents that enter the enterprise network, it is possible automatically to detect and drop all malformed messages. Whether the non-conforming XML document is the result of accidental bad configuration, network packet corruption or deliberate attack, intelligent validation greatly reduces the threat and notifies the network administrator of the problem. An administrator can then verify whether a configuration error or an attempted security breach has triggered the alert, and address it appropriately.

Today, many XML messaging systems (whether home-built, based on commercial SOAP stacks or otherwise) do not perform schema validation when in production mode. This is due to the fact that schema validation is so slow compared to simple parsing. Developers too often are forced to compromise on security for the sake of performance.

Tag-level protection: XML signature and encryption

Complex multi-party transactions and highly integrated supply chains require the ability to conceal and/or protect from modification XML documents or sections of documents. This is accomplished using tag-level XML encryption and XML public key signing, as specified in the XML Signature and XML Encryption specifications from the W3C.

The principles behind XML PKI and encryption are the same as those used elsewhere, but these technologies allow portions of an XML message to be modified or viewed while protecting other sections from reading and writing by select transaction participants. A user-defined tag can be:

- **selected from the message (using XPath)**
- **signed using a private key**
- **reinserted into the document.**

The recipient can than verify (using the signer's public key) that the tag, and its contents, were not modified in transit. For example, the message may be an order request, where all trusted parties are allowed to modify everything except the account number and the total amount. In this case, the <acctno> and <amount> tags — and their contents — could be signed by the sender to ensure that everyone could read them but no one else could modify them without being detected.

On the other hand, tag-level XML encryption would make it so that it was impossible to read the contents. Tag-level encryption works hand-in-hand with transport-level encryption. It represents one of the last security barriers.

The performance impact of secure XML

Finally, it is important to pause and consider that XML security processing is largely an XML processing problem. Complex cryptographic processing — coupled with XML parsing, filtering, validation and XPath/XSLT processing — are all required to perform the functions described above.

These are extremely resource-intensive. Unfortunately, they are sometimes simply turned off today in production systems — because they slow down the very transactions they are supposed to protect.

No e-business application should be giving up security due to performance concerns. Tomorrow's XML security engines must address XML security issues without creating performance bottlenecks.

Management conclusion

While it may be too early to tell what kind of new 'XML-aware' infrastructure or services will be most successful, XML developers should expect that network infrastructure a year from now will have new levels of intelligence. There will be purpose-built 'XML-aware' hardware for XML acceleration and security.

Without this, XML might yet be compromised by:

- ***either security failures (especially if these are shown to result from operations 'switching off' certain key aspects for performance reasons***
- ***any rising belief that XML means slow and cumbersome, and so should therefore be avoided in the interests of speed.***

Members of the International Advisory Board

Charles C.C. Brett
President, C3B Consulting Limited & President, Spectrum Reports

William Donner
Fenway Partners

Kathryn Dzubeck
Executive Vice President, Communications Network Architects, Inc.

Ellen M. Hancock

Paul Hessinger
Vision UnlimITed

Pierre Hessler
Deputy General Manager, Cap Gemini

Michael Killen
President, Killen & Associates, Inc.

Dale Kutnick
President, Meta Group, Inc.

Thomas Curran
Chief Technology Officer and EVP, Bertelsmann Media Worldwide

Norris van den Berg
General Partner, JMI Equity Fund, LP

Fiona A. Winn
Managing Editor & Publisher Spectrum Reports

Additional contributors include:

Francis X. Dzubeck
Communications Network Architects, Inc.

Jay H. Lang
Distributed Computing Professionals

Keith Jones
IBM

David McGoveran
Alternative Technologies

Will. Capelli
Giga Group

Amy Wohl
Wohl Associates

Martin Healey
Technology Concepts Limited

Mark Allcock
J.P. Morgan Asset management

Aurel Kleinerman
MITEM

Chris Cotton
Consultant

Ian Hugo
Year 2000 Taskforce

Yefim Natis
Gartner Group

Rosemary Rock-Evans
Consultant

Beth Gold-Bernstein
Hurwitz Group

Tom Heywood
University of Southampton

Eric Leach
ELM

Glen Macko & John Parodi
Digital Equipment Corporation

Randy Rhodes & Troy Terrell
Black & Veatch

Colin Osborne
The Tivyside Group

Roy Schulte
Gartner Group

Jim Johnson
Standish Group

Tom Curran
TC Management

Alfred Spector
IBM Corporation

Max Dolgicer
International Systems Group, Inc.

Peter Bye
Unisys Systems and Technology

Ely Eshel
MINT Communication Systems

Steve Ros-Talbot
SpiritSoft

Peter Houston
Microsoft Corporation

Jeff Tash
Database Decisions

Ed Cobb
BEA Systems

Bernard Abramson
Merck & Co.

Mirion Bearman and Kerry Raymond
CRC for Distributed Systems Technology

Geoff. Norman
Xephon

Jim Gray
Microsoft Research

Jason Longo
PRL Scotland

Wayne Duquaine
Grandview DB/DC Systems

Steve Craggs
Saint Consulting

Tom Welsh
Consultant

Gustavo Alonso
Swiss Federal Inst. of Technology

Mark Whitney
Delta Technologies

MIDDLEWARE**SPECTRA**
is published and distributed worldwide by:

USA and Canada:
Spectrum Reports, Inc.

Subscription Center
PO Box 32510,
Fridley, MN 55432, USA
Telephone: 763 502 8819
Fax: 763 571 8292

UK and Rest of the World:
Spectrum Reports Limited

Research and Editorial Office
St Swithun's Gate, Kingsgate Road
Winchester SO23 9QQ
England
Telephone: +44 1962 878333
Fax: +44 1962 878334

Subscription Centre
St Swithun's Gate
Kingsgate Road
Winchester SO23 9QQ
England
Telephone: +44 1962 878333
Fax: +44 1962 878334

Email and Internet

Email:
spectrum@middlewarespectra.com

World Wide Web:
www.middlewarespectra.com

ISSN 1356-9570

[incorporating FINANCIAL MIDDLEWARESPECTRA ISSN 1460-7220]