MIDDLEWARESPECTRA

incorporating FINANCIAL MIDDLEWARESPECTRA

Contents

February 2005

Volume 19 Report 1

Middleware's progress

Edward Cobb
Vice President of Architecture and Standards
BEA Systems

Management introduction

Edward Cobb is Vice President of Architecture and Standards at BEA Systems (San Jose, CA). At BEA he is responsible for co-ordinating and managing BEA's involvement with external standards organizations. In addition he has the challenge of managing BEA's co-operation and partnership arrangements with IBM and Microsoft regarding Web Services and the Java partnership with IBM.

Mr. Cobb joined BEA in 1997. Before this he worked at IBM, principally with IMS and CICS middleware. As such he is in an excellent position to discuss:

- ***the past evolution of middleware over the past five years, and more***
- ***where middleware is going in the future***
- ***how and why standards are playing an increasingly important role in the way that middleware is moving forward.***

The middleware arena 5 years ago

Five years ago was an interesting time for middleware, especially looking back now. It was about the 'high' point (if you can reasonably describe it like this) of the Internet/dot.com frenzy. At the time BEA was about 5 years old, having been founded in 1995. It purchased Tuxedo from Novell in 1996 and then WebLogic (with its Tengah Application Server) in 1998. It now had the foundations of today's middleware portfolio.

1999 was important for BEA for many reasons. It was the year that Application Server sales started to take off. BEA achieved sales of more than $464M for the year ending January 2000 and, by the end of the next year, sales had increased to a $1B annual run rate.

This was also the time when companies were looking for technologies to exploit the Internet. If you recall, Java had arrived about 3 years earlier and was, by 1999, in full flow. Initially it had been touted as a client-side technology. But by the late 1990s it had begun its transition towards server-side relevance, a move that was encouraged with the emergence of Application Servers plus the announcement of what is now called Java 2 Enterprise Edition (J2EE).

This was also about the time of the rift between Sun and Microsoft over Java — which caused Microsoft to go off and develop 'something different', which would become its own server platform, Microsoft .NET. A direct consequence of this decision was the subsequent popularity and high growth of Application Servers as exclusively a Java phenomenon, for a very long period of time.

A second dimension, that I think most now appreciate, was that the dot.com era was based too much on the promise of the Internet being combined with what 'might be'. Everybody knew that the Internet was important. The problem was that there was too much focus on what you might potentially use the Internet for in the future — rather than on how one could use the Internet to streamline what was already being done in businesses today.

Let me give an example. You will recall that there was lots of talk about global supply chains that would enable businesses to obtain optimum pricing from anywhere. The concept was that businesses would be able to collect the best possible deals from suppliers anywhere on the globe. This proved to be fanciful. The focus should have been on the gains that could be obtained just by utilizing the Internet as a way of streamlining what was already happening within an organization and among its existing suppliers.

Having failed dismally to deliver this grandiose vision there has been massive retraction. The past 3-4 years have been about more limited goals, concentrating on what can be achieved today, rather than striving to obtain utopia at some time in the future. The dot.com bust has proved to be a fundamental re-shaper of customer attitudes, and this applies across the whole middleware arena.

The Application Server

The original purpose of an Application Server was to enable users of the Internet to access applications and data already running on back end systems. In the beginning Application Servers were essentially about providing an Internet front end to existing applications and data.

This was a direct spin off from HTML, Web Servers and browsers. In 1999, probably the height of the browser experience, customer organizations were looking to enable millions of browsers to access the applications and data that they already possessed on existing host systems.

To begin with, Application Servers were used to enable a Web Server to:

- **pass a request to (say) a database**
- **receive the response back**
- **pass this back to the Web Server for display on the originating browser.**

This started out as a form of client/server, in much the same way as PC-based client/server developed — with the exception that there was not as rich or as heavy a client (in the sense that the client could have a large amount of programming on it). That particular constraint came from the limitations of network technology to support rich clients by downloading applets to the browser., a transient phenomenon that would be subsequently overcome — but not before the paradigm had shifted.

But what was really happening was the creation of a middle server tier. This tier between the actual browser and the applications or data at the back end performed the function that, in the traditional PC-based client/server world, might have otherwise been performed by a PC.

Subsequently (and this was similar to what happened with traditional client/server), developers discovered that you did not necessarily want to go directly to the database all the time, and that there were certain advantages to be gained if you put application functions between the browser and the data access layer. From this realization was born (or re-born) the Application Server — a Web-oriented version of the familiar transaction processing monitor.

This is, essentially, what today's Application Servers have become. They provide the transaction processing linkage between Web Servers (which serve browsers) and existing applications and data. They also provide a platform on which to write new applications (or services) that support the browser/Web Server combination in its own right.

More recently

As time progressed, even more changes emerged. First of all, I think as far as the Web itself is concerned, people began to lower their expectations. Rather than looking for the universal supply chain management solution that involved the entire known universe, developers started to consider how they might use the Web to make existing operations more efficient.

One result of this was that the notion of the 'big Internet' as being beyond and outside an organization's network began to wither. Instead the focus was increasingly on moving Internet technology closer inside each organization or extended enterprise.

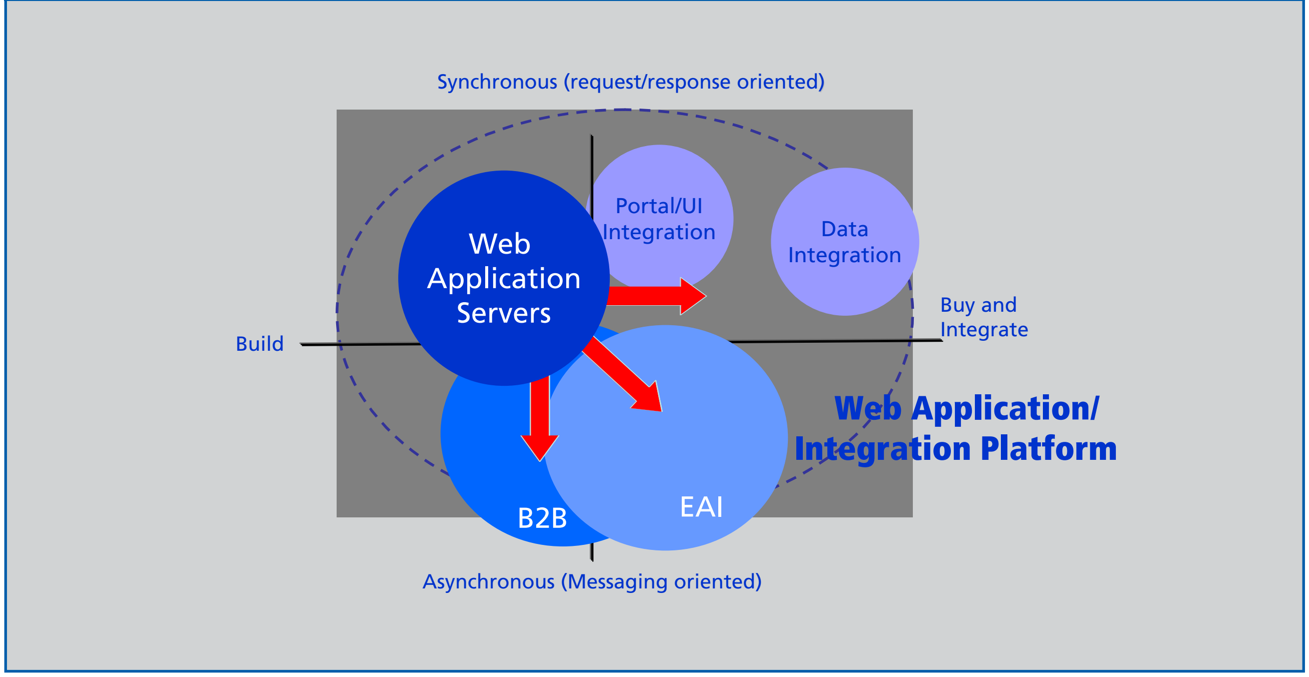
Applications shifted focus accordingly. Today you do not hear about 'open market buying' systems for global supply chains. Instead the discussion is narrower: it is about the specific supply chain which involves existing business partners or customers — people or organizations with whom you already have a contractual kind of relationship. It is no longer about the universe.

The second development was the realization that the existing massive infrastructure of systems, networks, applications and data that had grown up over many, many years, not only could not be replaced, but was a valuable source of business differentiation. Web-based technology was, in effect, 'just' another alternative or option that needed somehow to be tied together with all the existing IT investments. Thus the integration challenge re-raised its ugly head with a vengeance (in my view, integration has always been an issue, albeit one where the IT industry — both vendors and users — have never done a good job of addressing it adequately, much less solving it).

The Application Server needed to evolve again. All the new Java-based applications that had been developed to take advantage of the Internet had become yet another example of technology that needed to be integrated with the existing IT infrastructure at the back end (Figure 1.1).

In my analysis, integration never really declined in importance. After the dot.com crash, and with the renewed emphasis on business value over technology and using what already exists, integration has become ever more important. That said, I think one aspect that did change — from an integration perspective (and we are still in the

Figure 1.1: The Application Server integration dimension


Synchronous (request/response oriented)
Portal/UI Integration
Data Integration
Web Application Servers
Build
Buy and Integrate
Web Application/ Integration Platform
B2B
EAI
Asynchronous (Messaging oriented)

midst of this for not all have made the transition completely — is that integration is increasingly perceived as needing to be based on standardized technologies. Certainly a number of IT vendors, and BEA is one of the more prominent, have argued that the merits of using standardized technology offer more than the myriad of different proprietary technologies (that have been around for a long time) can do. The hope is that, in the not too distant future, there will be a market consolidation in the integration space similar to what has happened with Web Servers, Application Servers and databases.

From a business perspective I have long been puzzled why the integration market — which has been around for a long, long time — has not seen the consolidation you would expect to happen. There continue to be lots and lots of moderately important integration vendors who are either modestly successful or manage (marginally) to survive (usually from consulting rather than product revenues).

Why is this? I am not completely sure. Part of it has to do, I suspect, with the fact that integration is a hard problem to solve: it requires tying together so many different moving parts, most of which were designed to be standalone solutions.

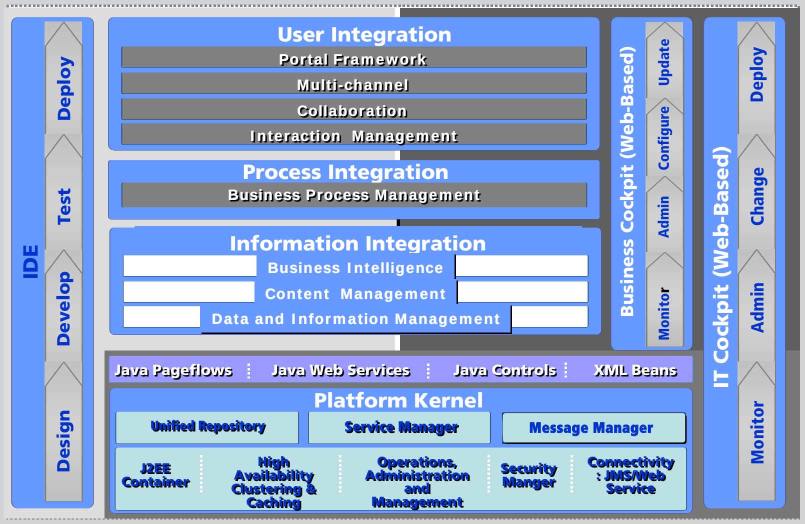
Furthermore, every organization has its own particular combination of different moving parts and none of the individual moving parts has yet to see sufficient motivation to standardize for the benefit of the other moving parts. Thus, SAP (for example) sells its applications based on solving a specific business problem, not how well those applications integrate with existing transaction systems, Application Servers or anything else.

In addition, integration has always required point solutions with lots of consultants or specialists to figure out how to tie existing applications into the latest version of SAP, or Oracle or PeopleSoft or whatever. The result has yet to see a common theme emerge.

Beyond integration, there has been consolidation, primarily driven by the dot.com bubble collapsing. One relatively recent example is what has happened in the Application Server space. There a few larger vendors — BEA, IBM and Oracle being the most prominent — have become bigger as organizations prefer to buy from the market leaders, with the new alternatives not being small commercial operations but rather Open Source Application Servers like JBoss, the Geronimo project (from Apache) or Tomcat.

The most import exception is Microsoft which, with .NET, embraces Application Server functions without offering an Application Server per se, in part because of its lack of Java support (which many view as a defining characteristic of an Application Server, even though there is no technical

Figure 1.2: The Portal model



reason for that to be true). As a result of its extensive investment in its own .NET technology, Microsoft is finally in an excellent position to sell its technology and products — particularly to departments and small or medium size enterprises. Interest in larger enterprises remains modest so far, but there is plenty of money to be made outside large enterprises.

What is now happening to the Application Server market is that the medium-sized vendors are being squeezed:

- **at the low end by the Open Source commodity software business**
- **in the middle by Microsoft**
- **at the high end by the likes of ourselves, IBM and Oracle.**

The next 2-5 years

My feeling is that the next 2-5 years are going to be concentrated around three particular technologies:

- **Portal (Figure 1.2)**
- **Process Management (Figure 1.3)**
- **Web Services and SOA (Figure 1.4).**

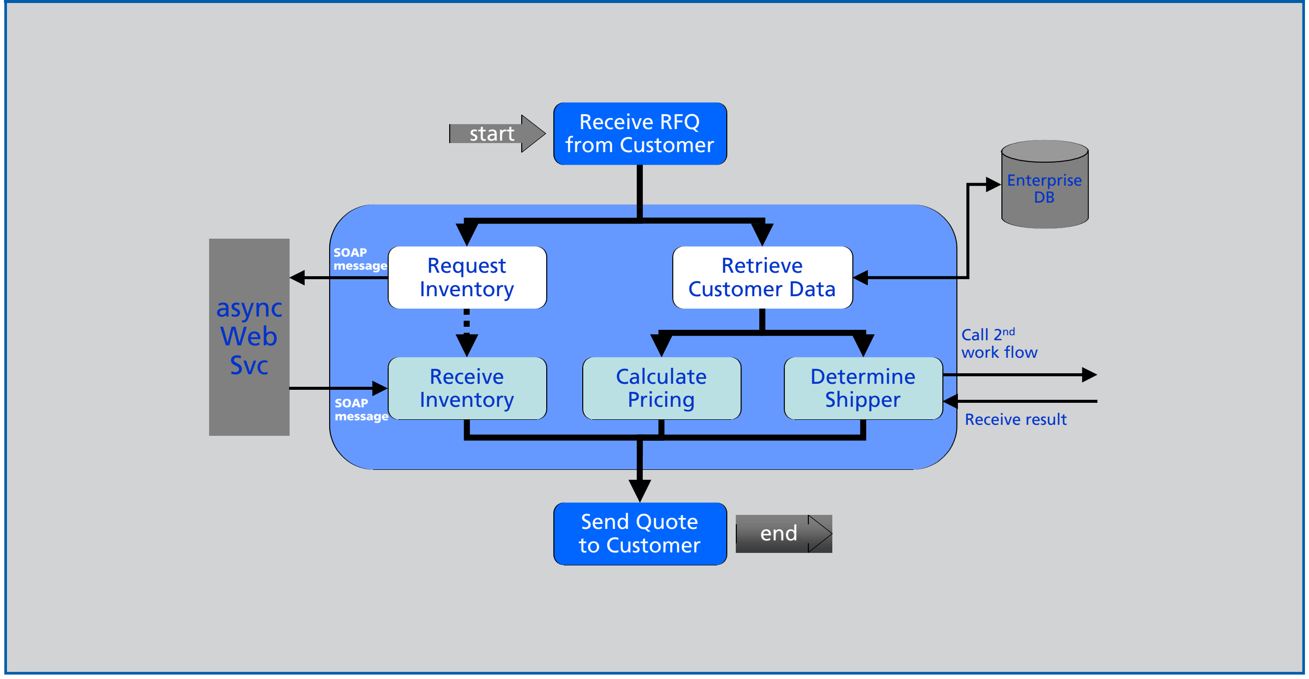
The most immediate of these, in my opinion, are portal technologies. Portals have really been the success story of Web-based applications, both within and outside organizations. The 'self service portal' — whether it is an internal HR portal for employees or a customer-facing portal for those interacting with the business — appeals to nearly all organizations. Portals now represent probably the largest and some of the most successful Web-based applications.

Is there any unique technology behind a portal? Some would argue not. But my view is that there are a couple of characteristics to consider. A portal exploits the Web browser for the user interface. You could even argue that a portal is a process technology specifically designed for screens — irrespective of whether these are traditional 3270 mainframe ones or modern browsers. As part of this you need a unique kind of flow control, enabling the required pieces of many applications to come together on the screen fairly easily.

In addition, a portal allows you to take the difficult back end integration problem and present the end user with a solution that assists the user visually to integrate the pieces — even if there is no real integration at the back end(s). When a user has his or her portal, the details on the screen arrive from multiple different applications. The user now can aggregate all the information needed on the screen to be able to do his or her job.

To me portals are 'integration on the glass' or front-end

Figure 1.3: Process Management



integration. Traditional integration occurs at the back end, between applications.

Moving on in time, I do think that process technology is also going to make a big difference. Why? Because the big problem that people have had whenever they have tried to build mega-applications is that you end up putting them together with 'spit and baling wire' and there isn't a lot of help in terms of tools or easily-understood technologies.

The idea, therefore, of work flow (which is not new) makes sense, especially if the work flow is not about documents (which is where work flow originated) but about orchestrating business processes. What work flow has brought to the table are reasonable tools for people to understand the actual business processes that underpin their organizations.

The key to me is to separate the business process logic from the IT logic that implements a specific business component. This is one of the ways that a Service Oriented Architecture (SOA) can help. If we do this, I feel there will be a real return on investment for those organizations that are prepared to take the time to understand their business processes and then how these processes might be architected using an external 'process manager' and re-usable business components, implemented as services.

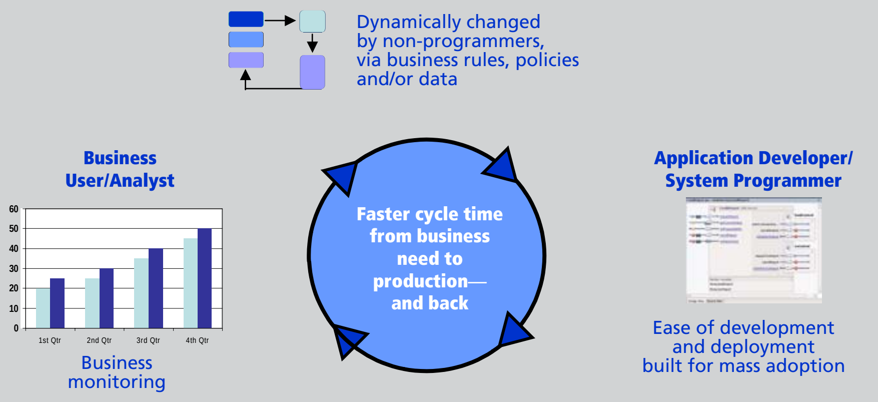
Finally, the idea behind Web Services, as a technology for implementing SOA, is simple, elegant and certainly not new. The Internet may be new but Web Service technology also has many things going for it that its earlier predecessors (DCOM, CORBA, DCE and others) lacked.

First, from a technical point of view, Web Services recognized that the right way to optimize communications was between enterprises — not just within each enterprise. With this came the realization that the optimal communication style was not RPC; but rather that it was messaging. Messaging technology, which is inherently based on loose-coupling, is much less brittle as well as more amenable to change, and vastly more flexible.

To me, accepting these two thoughts will have a real impact — because the effect is to architect for change up front, which makes everything more adaptable to change as time goes by. To support my thinking, just look at the Internet. One of the reasons it has been so successful: it is that it is very adaptable to change. In the absence of central control, no other approach could have succeeded.

That is not to say that all is rosy. The downside of message-based application programming is that it is hard. If you look at JMS or MQSeries or any other messaging engine, these are highly efficient and do a great job when they are running. But the current state of the art is they are not simple to program.

Figure 1.4: The Web Service goal



Production Environment

- Integrated with legacy systems through standard interfaces, in a Services Oriented Architecture
- Unified management and security
- Scale in performance and reliability based on demand

Why is this? One factor undoubtedly is that most of today's application developers were brought up to use tightly-coupled, synchronous programming models. This is not surprising. Arguably, it is inherently simpler to understand subroutines (the genesis of RPC), which are an essential element of nearly all programming languages.

But the result is that we have a pre-established mindset one which understands what a subroutine is and what it can do. In contrast, the messaging paradigm is not one that most people encounter when first learning a programming language. Since most developers begin with a programming language, this naturally predisposes them to the tightly-coupled approach.

Indeed, I would suggest that there is a real opportunity for somebody to figure out how to support messaging with the kind of tooling that is available for tightly-coupled solutions. If this can be done in as simple a way that Visual Basic transformed client/server programming, corporate developers will be able to deliver loosely-coupled designs, supporting applications that are loosely-coupled and which utilize messaging technology underneath. BEA's WebLogic Workshop is an example of a development tool which is targeted to building these types of applications.

We would then have the best of both worlds. With loosely-coupled designs you gain an architecture that scales as well as one that is amenable to change. From an implementation point of view, if tooling supports the loosely-coupled model adequately, you have the ability to build applications more quickly using existing knowledge. Furthermore, you are far more likely to achieve re-use as well as many of the other kinds of benefits that IT people have been discussing for years.

Essentially, this is the technical promise of Web Services. But there is more. Web Services do not possess the open conflict in the market that was so clear when CORBA was trying to win acceptance and Microsoft espoused a competing technology (COM). Then Sun introduced Java, as the 'only needed programming language and a much easier solution'.This was not exactly a recipe for ubiquity.

Web Services seems to be an exception in the evolution of software technology. It has today or, at least appears to have, all the major players behind its fundamental concepts and these are committed to making it a success.

Nevertheless, the problems that Web Services are trying to solve are truly complex. There are many different technologies involved as well as multiple different vendors engaged plus more than 20 specifications at last count, most of which have yet to be delivered to a standards body. This helps to explain why the standardization process is going so slowly.

It is certainly true we are not there yet. But, from my perspective of managing BEA's Web Services initiatives and support, I do believe we will get there — later rather than sooner but with some spectacularly complicated issues resolved for the long term.

Standards

Standards matter. They have always mattered. But in my role at BEA, I see them as mattering ever more — for vendors and customers. This is a new reality that, I think, almost every organization involved with IT recognizes.

Is this a major change? Sometimes, although this depends largely on the organization involved. But increasingly I see that the most fundamental disagreements are about which standards you should have, rather than about the principle of standards.

And there are other non-technical reasons to believe that standards really matter. I cannot think of a single major IT vendor that is not participating in the standardization process, even those who have not done so in the past.There is a growing recognition that, in order to be successful in the enterprise, standardization is something we must all pay attention to. This is equally true for both the large and small, regardless of one's current position in the marketplace for, as we have seen in the past, the status quo does not remain that way forever. It is all too painfully possible to be on the inside today and on the outside tomorrow.

While it is good news that almost everybody is engaged, that could also be bad news — because it means there are many differing, sometimes conflicting objectives. Standards are necessarily both an economic-political process as much as a technical one. Everybody has an agenda which they are trying to manage in a way that gains them maximum advantage.

One result of this has been the proliferation of standards organizations. This I do not consider to be good. In fact it is particularly frustrating — and almost every day you hear of yet another one. In practice this means we have to be highly selective in deciding which organizations are most likely to be important and will help us achieve our business objectives — and which ones are not going to be relevant. This decision usually needs to be taken long before the 'right' answer is obvious.

Deciding among them, therefore, is not simple. There are so many possibilities — and so few clear boundaries between them. One of the reasons there are few clear boundaries is that none of these organizations knows how to recognize their job is complete, declare victory — and then abolish themselves. Instead, with their original mission accomplished, they seek out new problems to keep themselves going. If only they would adopt a policy of self-redundancy.

The net result is that the old organizations continue and new ones just keep getting created. This is particularly wasteful of very scarce, but very good, talent.

Lessons learned

One of the aspects most different about working at a company the size of BEA and working at a giant like IBM is that you can't do everything. BEA — being 'only' a $1B company — needs to be much more selective about what it chooses to do and it has very little opportunity to 'cover its bets' by engaging in competing alternatives.

When you work in a company the size of IBM, you can engage in almost everything — and you can reasonably expect that you will be engaged in the right things because all your bases can be covered. That means that you can afford mistakes and you expect to make them. When you are smaller, the imperative to place bets on the right technologies and on the correct standards activities is much greater. In addition, you have to be right most of the time — because you do not have the resources to cover the competing alternatives.

So one of the most important lessons I have learned is to be much more selective in what you decide to do. You need to think hard about:

- **what your agenda really is**
- **how best to accomplish it**
- **how to leverage partnerships**
- **how to work with others in the industry.**

One must also acquire a thick skin. This has two facets, to:

- **deal with those who refuse to understand that you cannot do everything**
- **have a sense for when to change position if the 'market winds' change.**

My second lesson learned has come directly from customers. It is obvious, once you think about it, that customers are interested in solutions that work and solve their business problems. This keeps getting reinforced as we go forward. Technology is a means to an end, not the end in itself — a concept that is often difficult for technical people to accept. Increasingly I see us as needing to couch the solutions we wish to offer, and the decisions that we are required to make, in terms of the specific business problems our customers are facing.

The third lesson that I have learned from my involvement with BEA in dealing with customers is the importance of partnerships. BEA is not a one-stop shop and never will be a one-stop shop. However, customers still need complete solutions. This means that we have to work with others in our industry if we are to meet our customers' expectations (and thereby be successful in the marketplace).

There is no doubt in my mind that success today comes not only from being able to deliver our own technology effectively but also from being able to combine what we have with what others have to create a complete experience — from technology through marketing, support and sales — to satisfy what the customers want and need. This may not be original but it matters more and more in today's middleware arena.

Management conclusion

During a long career in IT, Mr Cobb has seen many developments across many platforms. This gives him a most useful perspective when working with middleware and Web Services. He possesses the context to see what is constructive and new — and what may be re-inventing the wheel.

In this discussion, he has covered how middleware has evolved from uncertain beginnings towards Web Services. That these Web Services are another means to deliver integration of existing application systems is a dimension that many forget, or choose to forget. Yet this is the requirement which must be satisfied if Web Services are to realize the aspirations that customers have.

Web Services standards process: coming off the rails?

Tom Welsh
Consultant

Management Introduction

Since the advent of Web Services four years ago, it appears that IBM and Microsoft have found a way of circumventing the standards consortia. Neither can afford, or are able, to lay down its own proprietary 'standards' any more. But both of them, acting jointly, may be able to do so.

In the past three years, IBM and Microsoft have published dozens of WS- specifications, which all fit into a grand Web Services architectural vision. There is only one small problem: that vision is not endorsed by any standards body. It is beginning to look as if IBM and Microsoft are setting up a quasi-duopoly, at least as regards Web Services. No other vendors — and certainly no users — have much say when it comes to making critical decisions about these specifications.*

As Tom Welsh discusses, there are serious risks to this approach. Too many specifications are being created too quickly, and the dependencies between many of them have not been fully understood. As a result, a whole new consortium — the Web Services Interoperability Organization (WS-I) — has had to be established, to paper over the many cracks in the multitude of WS- specifications.*

Boring but useful: software standards consortia

How true it is that 'familiarity breeds contempt'. We have all heard of the bad old days when computer manufacturers created their own rules, like so many giant dinosaurs stamping through the steaming jungles of the Old Computing Age. But who feels gratitude or appreciation for the open standards that, today, protect us from a return to those days?

Speaking on World Standards Day (14 October 2003), European Union Commissioner Erkki Liikanen made the case for standards in the following words.

"*Open standards are important to help create interoperable and affordable solutions for everybody. They also promote competition by setting up a technical playing field that is level to all market players. This means lower costs for enterprises and, ultimately, the consumer*."

Fair access to the marketplace, and lower costs all round — how can that be bad? Yet open standards for software did not become commonplace until about 20 years ago. As recently as 1970 the IT industry consisted essentially of 'IBM and the seven dwarves' — Burroughs, Control Data, General Electric, Honeywell, RCA, Scientific Data Systems and Univac. Each company had its own installed base of customers, who had invested heavily in that particular vendor's hardware and could scarcely afford to contemplate changing suppliers.

At that time, software was barely on the brink of becoming an independent industry. Operating systems, database management systems and other 'packages' came from the hardware vendors. Of course, each vendor's software was entirely incompatible with anyone else's hardware (or software).

Even in those early days, there were some standards bodies. But these were not dedicated to software — or even to computing. When the precursor to the American National Standards Institute (ANSI) was founded in 1918, one of its first initiatives concerned the standardization of pipe threads. Later, after the Second World War, it joined with the standards bodies of 25 other countries to set up the International Organization for Standardization (ISO). ANSI and ISO were responsible for the first wave of global, vendor-neutral software standards: those for programming languages included FORTRAN, COBOL and, later, SQL.

For whatever reason, the 1980s witnessed a rapid increase in the number of standards consortia. Perhaps it was stimulated by the appearance of UNIX and associated specifications, such as those for the Internet protocols. Previously, if you bought a computer you had to get the associated operating system from that computer's manufacturer. By 1980, UNIX was becoming a free or low-cost alternative — and it ran on more and more types of hardware.

This led to the infamous 'UNIX wars' of the late 1980s, in which groups of manufacturers ganged up in rival cliques, each of which attempted to establish its brand of UNIX as the global standard. This debacle is still blamed for UNIX's failure to become as dominant as many had expected. As much as anything, the UNIX wars forced vendors to admit to themselves that it might be better — in Benjamin Franklin's words — to hang together rather than to hang separately.

This gradually led to a grudging acceptance of the benefits of standardization and many standards 'bodies', including:

- **the European Computer Manufacturers' Association (ECMA), which dates back to 1961**
- **X/Open, which was founded in 1984**
- **the Internet Engineering Task Force (IETF), founded in 1986**
- **the Object Management Group (OMG), founded in 1989**
- **the Organization for the Advancement of Structured Information Standards (OASIS), founded in 1993**
- **the World Wide Web Consortium (W3C), founded in 1994**
- **the Java Community Process (JCP), founded in 1998**

to name but some.

Even while signing up for membership of standards consortia, some vendors seemed to have their fingers crossed. They co-operated reluctantly, seemingly always ready to break away on their own if commercial advantage beckoned. Even their customers, who might seem to have most to gain from vendor-neutral standards, did not always see things that way. Carl Cargill, author of a 1989 book on standardization, described an all-too-common reaction:

"*To complicate matters further, in IT standardization, users rarely participate on standards committees and are virtually absent from the accredited [standards-developing organizations]. The cost of participation is prohibitive for most users but, perhaps more importantly, users tend to be less focused on long-term IT standards than they are on the use of standards to provide solutions for business problems, both current and future. As a result, they feel that the activ-*

ities of the standards committees have little relevance for them."

To make matters worse, the big vendors — and especially market leaders — tended to dislike open standards. In their case, the problem lay in the specific effects highlighted by Liikanen: create a level playing field and you lower costs for users. But the whole point of being a market leader is that you are trying to keep the playing field tilted in your favor, enabling you to continue charge higher prices.

Like everything else in engineering, IT standards involve trade-offs:

- **on the one hand they lower barriers to entry, promote competition and reduce the costs that users have to pay**
- **on the other hand (in addition to exasperating leading vendors), they tend to slow down the pace of innovation, are open to all the usual criticisms levelled against committees and are extremely expensive (for vendors) to support.**

One proprietary vendor bad; two proprietary vendors good?

Among the proprietary 'dinosaurs', IBM stood head and shoulders above all others. From its System/360 hardware and OS/360 operating system to CICS, IMS and SNA, it was (and still is) the quintessential proprietary vendor. Indeed, one of its greatest practical problems arises from its very success in laying down and promoting proprietary de facto 'standards'. After only a couple of decades, there were too many of them.

By the 1980s IBM found itself embarking on an apparently hopeless search for the Holy Grail of transparent interoperation between its many diverse operating systems and other software platforms. This was known as SAA (for Systems Application Architecture), where the objective was to harness IBM's own disparate offerings — from MVS to VM to OS/400 to OS/2 and related elements.

Microsoft, which eventually became second only to IBM — before overtaking it (IBM) in the realm of software — was more exposed when it came to the issue of respecting open standards. IBM at least had the excuse that most of its business was in hardware, which everyone expected to be proprietary. Moreover, it had been around for decades, and therefore had a lot of 'legacy' architectures which were irretrievably proprietary. In comparison, Microsoft specialized in software and, although founded in 1974, only entered the limelight after 1980 when it became the single source for MS-DOS, the operating system for what was then known as the IBM PC.

DOS was a success; its successor (Windows) eventually became one too, after many years in the wilderness. Then along came Word and Excel, and the relatively robust Windows NT — each, in its own way, demonstrating Microsoft's remarkable ability to learn from others and then, what is more, to profit from that learning.

What Microsoft had come up with, astute observers soon noticed, was a strategy for embracing and extending other people's ideas — whether these took the form of software products or open standards. In due course, and from a position of strength, this strategy evolved into what some described as "embrace, extend and extinguish" (or EEE). Wikipedia, the online encyclopedia, describes EEE pithily:

"The three stages of the EEE strategy appear to be unfolding as follows:

- **"Embrace: Microsoft publicly announces that they are (sic) going to support a standard. They assign employees to work with the standards bodies, such as the W3C and the IETF.**
- **"Extend: They do support the standard, at least partially, but start adding Microsoft-only extensions of the standard to their products. They argue that they are trying only to add value for their customers, who want them to provide these features.**
- **"Extinguish: Through various means, such as driving use of their extended standard through their server products and developer tools, they increase use of the proprietary extensions to the point that competitors who do not follow the Microsoft version of the standard cannot compete. Unfortunately, the Microsoft version uses proprietary technologies such as ActiveX that places competitors at a distinct disadvantage. The Microsoft standard then becomes the only standard that matters in practical terms, because it allows the company to control the industry by controlling the standard."**

Many specific examples have been adduced; among the most prominent are:

- **HTML**
- **JavaScript**
- **Kerberos**
- **the Basic and C++ languages**
- **SMB networking**

- **Active Directory.**

There is even evidence that Microsoft was in the early stages of an EEE manoeuvre against Java when Sun overturned the board by launching a lawsuit.

If an industry watcher had been told in, say, 1999 that IBM and Microsoft would soon be collaborating closely on a major software initiative, the natural reaction would have been sheer disbelief. Throughout the 1990s, IBM never gave up struggling against Microsoft. When it finally became clear that OS/2 was doomed, it ported its best development products to Windows in the hope of winning the battle at that level. As soon as Linux and Java emerged as credible platforms, IBM seized on them and deployed them as weapons in the war against Microsoft.

When Dave Winer and Don Box brought the raw idea of Web Services to Microsoft in 1998, they hardly expected to get a hearing at all. Instead, Microsoft grabbed the ball and ran with it.

By early 2000, Web Services were at the heart of the company's brand-new .NET strategy. That is when IBM surprised the industry by joining Microsoft, instead of lining up against it.

In May 2000 IBM (and Lotus, by now a division of IBM Software) joined Microsoft, DevelopMentor and UserLand in submitting SOAP 1.1 — the first Web service specification — to W3C.

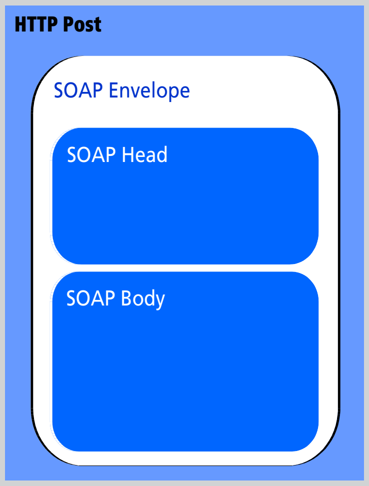
Indeed, if you look at the first ten Web Service specifications and the companies that wrote them, you obtain a list that looks like this:

- **SOAP: IBM, Microsoft, Lotus, DevelopMentor and UserLand (Figure 2.1)**
- **UDDI: IBM, Microsoft and Ariba (Figure 2.2)**
- **WSDL: IBM and Microsoft**
- **WS-Inspection: IBM and Microsoft**
- **WS-Security: IBM, Microsoft and VeriSign**
- **WS-Coordination: IBM, Microsoft and BEA**
- **WS-Transaction: IBM, Microsoft and BEA**
- **BPEL4WS: IBM, Microsoft and BEA**
- **WS-Reliable Messaging: IBM, Microsoft, BEA and TIBCO**
- **WS-Addressing: IBM, Microsoft and BEA.**

A pattern emerging?

Can you see a pattern emerging? Sometimes IBM and Microsoft publish specifications jointly, with no other contributors. Sometimes there are one or more others — usually companies with a special expertise (and reputation) in the relevant field. Thus, Ariba — an e-commerce pioneer — was associated with UDDI, a specification for global directories that would be necessary for e-commerce.

Figure 2.1: SOAP



Likewise, VeriSign was brought in to co-author the core security specification, and BEA and TIBCO for those related to transaction processing and reliable messaging.

In a still more ominous pattern of behavior, IBM and Microsoft seemed in little hurry to submit their specifications to standards bodies. In the early days they did give SOAP and WSDL to W3C — which took the responsibility seriously and plunged into a three year long period of study and discussion before adopting SOAP 1.2 as a Recommendation in June 2003.

Whether concerned about the delays, or unhappy with W3C's desire to tackle the associated design problems at a fundamental level, IBM and Microsoft then turned sharply away from it. UDDI was initially given its own organization, before gravitating to OASIS — where it was soon followed by BPEL and WS-Security. Recently, however, the Big Two have returned to W3C, submitting WS-Addressing for its consideration.

It soon became evident that IBM and Microsoft had a new policy for dealing with standards bodies. Instead of taking a problem to a consortium and setting up mechanisms for analyzing the problem and devising suitable standards, they have taken to publishing their own joint specifications and revising them repeatedly. Giving a specification to a standards consortium now looks more and more like a final, perfunctory rubber stamping of work that they (IBM and Microsoft) deem to be already essentially complete.

While IBM, Microsoft and their partners claimed that they could not afford to wait for standards bodies to deliberate at length — and even cited the wishes of customers in support of their bilateral initiative — it is clear that other motives might be at work. For example, the longer they kept full control of a specification, the more time they would have to develop products that would support it.

Consider this extract from the April 2002 paper "Business Process Standards For Web Services" by David O'Riordan, co-founder and Chief Architect of Bind Systems. "*Microsoft and IBM are clearly moving towards a set of specifications that would address both B2B and EAI requirements. It has been widely speculated that they will collaborate to produce a single proposal or set of proposals in this space that could then be submitted to the W3C for inclusion in its Web Services architecture stack in the process layer.*

"*There are, however, significant obstacles to be overcome for this to happen. Technical obstacles include the different approaches to control flow modeling (in XLANG control flow is described using a block-structured approach best represented graphically using flow charts, while WSFL uses a state-transition approach best represented graphically using UML activity or state graphs). This is not just an argument about the technical merits of the respective approaches — both vendors have significant investments in these technologies in their respective product lines (WebSphere from IBM and BizTalk from Microsoft).*"

Naturally, other vendors took a dim view of this nascent quasi-duopoly. Presently, some of them began publishing their own specifications. Almost inevitably these usually clashed with those of IBM and Microsoft. It was difficult, however, for any rival group to muster enough credibility seriously to challenge the world's two leading software vendors when these two were working together.

One example suffices. Fujitsu, Oracle, Sun, Hitachi, NEC and Sonic published WS-Reliability in January 2003, and submitted it to a newly-formed OASIS Technical Committee in February that year. IBM and Microsoft had apparently been out-manoeuvred this time. Not for long, however: in March 2003, together with BEA and TIBCO, they published their own WS-Reliable Messaging, which covered the same ground as WS-Reliability but in a completely incompatible way. Thus, users of reliable Web Services would be able to use either WS-Reliability or WS-Reliable Messaging — but not both.

It had seemed as if Fujitsu, Oracle and their partners had the upper hand, as their specification had been accepted by OASIS. This impression was reinforced recently, when WS-Reliability was formally adopted as an OASIS Open Standard.

However, IBM and Microsoft still refuse to accept it. The reasoning behind this peculiar decision was explained in a Microsoft document that was sent to various supporters of WS-Reliability — in mid-2003: "*A core requirement that drives the WS-ReliableMessaging specification is maintaining architectural cohesion within the specification and in relation to other Web Services specifications (WS-Security, Policy, and so on) and composability with other specifications that describe assurances (for example WS-Transactions). It is therefore very hard to proceed on final design of any one particular specification without commensurate progress on the others. Separating this specification's process from the other Web Services specifications it composes with would harm the goals of composability and architectural coherence. A litmus test for the web services architecture is bone fide interoperability and composability demonstrated between various implementations from several vendors.*"

Microsoft and IBM believe that the WS-* stack that they have constructed must be the starting point for any new work. For this reason, they are prepared to shun the proceedings of any standards body that does not accept their whole WS-* stack — past, present and future.

Meanwhile, the OASIS Web Services Reliable Messaging Technical Committee (WSRM-TC) takes the view that it should not build in dependencies on proprietary specifications (defined as those not submitted to, or created within, a standards body). Unfortunately, not only WS-Reliable Messaging but several of IBM and Microsoft's other specifications on which it depends, fall into this category.

The supporters of WS-Reliability and WS-Reliable Messaging have thus fallen into a classic deadlock:

- **the former have had their specification blessed as an industry standard, although it is incompatible with the framework proposed by IBM and Microsoft**
- **the latter rely on their enormous prestige and commercial influence to outweigh the authority of an OASIS standard.**

The strange tale of the Web Services Interoperability Organization (WS-I)

By the end of 2001 there were dozens of SOAP products. Developers were becoming concerned about interoperability. Even the major toolkits — like those from Apache, IBM and Microsoft — did not always see eye to eye.

When experts began drawing up cookbooks to help developers write interoperable Web Services, some common advice emerged. For instance, for two implementations to interoperate they would have to agree on:

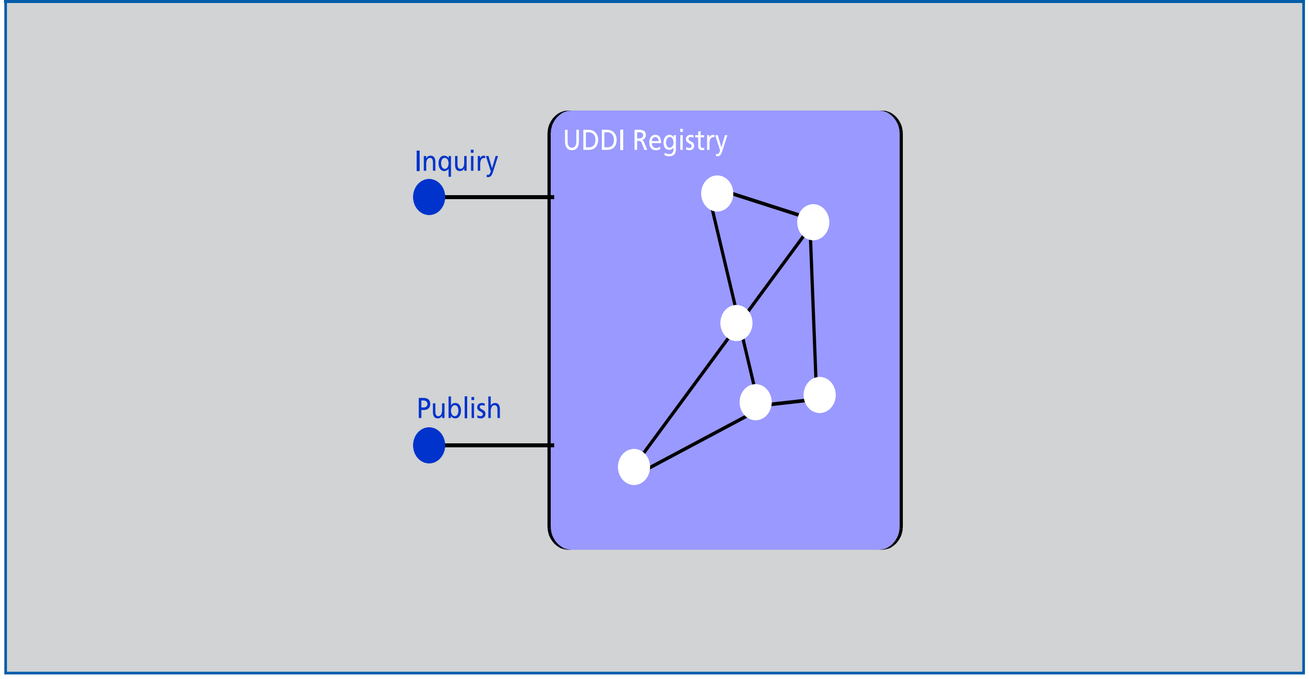
- **a common wire transport**
- **the same version of XML Schema**
- **the same version of SOAP**
- **the same version of WSDL**
- **the same message format.**

There were literally scores of ways to break interoperability. Furthermore, many of these were poorly (if at all) documented.

To address this problem methodically, the Web Services Interoperability Organization (WS-I) was founded in February 2002 by:

- **Accenture**
- **BEA**
- **Fujitsu**
- **HP**
- **IBM**
- **Intel**

Figure 2.2: UDDI



- **Microsoft**
- **Oracle**
- **SAP.**

These nine companies automatically became board members. Sun, however, was not invited to join the WS-I board, precipitating a scandal that was to last for months. A widespread perception arose that IBM and Microsoft wanted to run things their way, and neither of them saw any point in having Sun around to confuse the issue with its (often rather different) ideas.

WS-I was chartered “to promote Web Services interoperability across platforms, applications, and programming languages” (Figure 2.3). This might come as a surprise to those who thought that the whole point of Web Services was to guarantee interoperability out of the box. Explicitly denying that it was a standards body of any kind, WS-I declined to introduce a certification program. That would be too costly and slow.

Instead, it set out to provide profiles, sample implementations and guidelines. In short, it is the first industry consortium ever set up exclusively to review the work of other industry consortia and recommend ways of using them safely.

In its first 33 months of existence, WS-I has produced:

- **a Basic Profile**
- **a working draft of a Basic Security Profile**
- **a couple of ancillary Profiles**
- **a set of sample applications and testing tools.**

The Profiles contain much useful guidance about how to implement inter-operable Web Services. But they only deal with a few of the oldest core specifications — such as SOAP, WSDL, UDDI and Web Services Security in its most basic form. Moreover, they inevitably lag the publication of the latest specifications by a minimum period, which experience shows to be (typically) more than a year.

Since the foundation of WS-I, Tom Glover of IBM has been president and chairman, and Christopher Kurt of Microsoft secretary. It is unclear what — if any — provision there is for changing the holders of these posts. While they remain as they are today, it will be difficult to avoid the impression that WS-I, too, is controlled or (at least) strongly influenced by IBM and Microsoft.

Management conclusion

Vendor representatives have long had a field day criticizing standards bodies. To hear them talk, you would think that the consortia are too slow, too academic — and not nearly responsive enough to the commercial requirements of vendors.

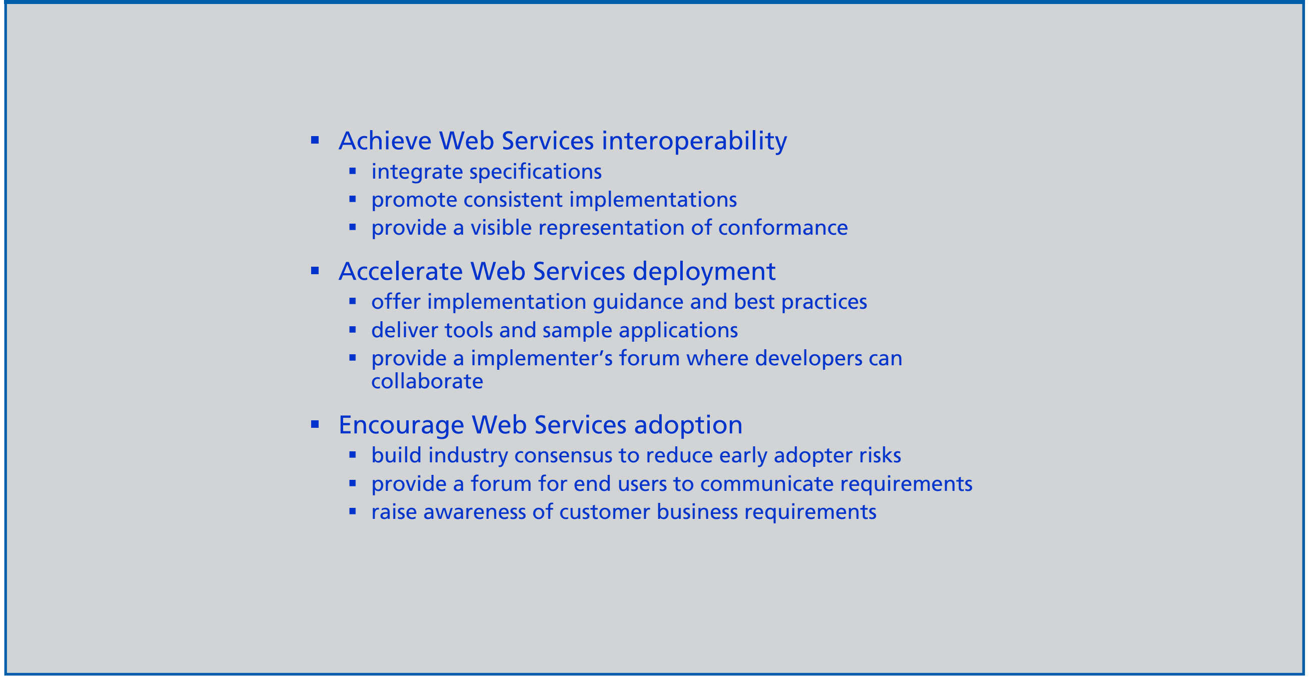


Figure 2.3: WSI goals

- Achieve Web Services interoperability
  - integrate specifications
  - promote consistent implementations
  - provide a visible representation of conformance
- Accelerate Web Services deployment
  - offer implementation guidance and best practices
  - deliver tools and sample applications
  - provide a implementer’s forum where developers can collaborate
- Encourage Web Services adoption
  - build industry consensus to reduce early adopter risks
  - provide a forum for end users to communicate requirements
  - raise awareness of customer business requirements

But these complaints are ill-founded. True, the open standards process has many defects and drawbacks. But it is still the best way of making reliable progress in the long run.

Winston Churchill once remarked that "No-one pretends that democracy is perfect or all-wise. Indeed, it has been said that democracy is the worst form of Government except all those other forms that have been tried from time to time". Much the same could be said about the open standards process. Whatever its weaknesses, it is steadier and more reliable than the alternatives.

There is an inevitable difference between the interests of any one company and those of the industry as a whole — not to mention those of users world-wide. IBM and Microsoft are naturally tempted to rush the creation of a system of specifications, so that they can start selling Web Service software and services on a grand scale. At the same time, they can hardly resist the temptation of giving themselves a little advantage over their competitors. It seems harmless enough; and after all, who can stop them?

The trouble is that haste really does make waste. The world's leading software companies should know by now that software development — and especially software architecture — cannot be rushed. If it is, things will go wrong; incompatibilities will open up, and eventually everything may take even longer than the standards consortia would have needed to do a thorough job.

Making the mainframe a Web Services peer player

Mark Lillycrop
Principal
Arcati Research

Management introduction

Despite the great enthusiasm for Web Services and so-called Service Oriented Architectures (SOAs), it is still true that most large organizations will not move far without continuing to use the mainframe data and processes on which their businesses depend for survival. Consequently, the use of standardized middleware to enable mainframe systems to interact with Web-facing applications is absolutely essential for the future development of many Fortune 500 companies.

In this analysis, Mark Lillycrop:

- ***looks at how mainframes have refused to die, because they have become 'open'***
- ***examines how this has been achieved, by using IBM's CICS and its Web Services participation as an example.***

Mainframes refuse to die away

There have been those who have predicted the demise of the mainframe. Indeed this occurred many times over the last fifteen years. Each time, so far, such predictions have been proved wrong (so long as you primarily contemplate an IBM mainframe).

Rather than sinking irrevocably into the corporate background, and waiting to be relieved of its workload by newer and more nimble technologies, the IBM mainframe has continued to prove that it is irreplaceable, at least for the foreseeable future. Its capability as a highly scaleable, highly available transaction processing engine remains to be challenged. Its role as the central repository of key corporate data is equally secure — with anything up to 80% of business-critical data still residing under MVS (or OS/390 or z/OS) control.

One of the problems the industry has faced with the mainframe is that it has remained resolutely distinct from every other platform that has come along since. Mainframes were always intended to:

- **run complex mixes of workloads, many of these being batch-oriented**
- **offer very high utilization rates**
- **deliver particularly good performance for transaction-oriented environments.**

Distributed servers tend to offer much better performance for numerically intensive work and single applications. They also ‘enjoy’ dramatically lower utilization rates. As such, their value proposition is so fundamentally different that industry pundits (and, indeed, IT vendors and users) might be said to have spent the last decade arguing about the relative cost of ownership of chalk and cheese.

Mainframes are open; their users might not be

For many years now, the mainframe has been an open platform as well as an early adopter of open standards (this applies, for example as much to Unisys mainframes as much as to IBM ones). But with their distinctive skill set and complex architectures, many mainframe users have tended to position the platform as a separate technical entity — a back end server in a world increasingly governed by Web Services and peer connectivity.

Having said that, it is definitely time to draw a distinction between:

- **legacy mainframe users (those who continue to rely on the system, either internally or via an external service company, for back-end data and processing — but who see no prospect of new applications for those large systems platform)**
- **strategic mainframe users (those who wish to leverage the mainframe’s strengths to build new Web-facing business applications).**

In both cases, it is essential for the mainframe platform to offer some level of support for key Web Service middleware. But the level of interoperability required — and the ease of Web-facing development — will vary considerably between these two groups.

For the ‘strategic’ group in particular, there is a strong incentive to make the mainframe a peer player in the Web Services arena (Figure 3.1) — so that new applications can be created fast and positioned according to a rapidly changing set of business criteria. This means that the mainframe must be capable of requesting Web Services as well as serving its data and transaction function. Without this two way flow, users who see real benefit in leveraging the strengths of the mainframe for new systems may face major technical obstacles in making it happen.

CICS and Web Services

The success of the IBM mainframe in the past, and the key to its future role in the large enterprise, comes down in no small part to CICS — the ubiquitous transaction processing monitor. CICS reportedly processes some 30 billion transactions a day worldwide, which is not bad for a piece of software that has just celebrated its 35th birthday.

Much of IBM’s focus recently (and that of supportive ISVs) has been on making CICS ready for Web Services. This has been achieved by developing its interoperability through the introduction of appropriate middleware components, particularly within the XML area.

The recent announcement of CICS Transaction Server for z/OS V3.1 seeks to position CICS much more favorably within a Web Services environment. This announcement focused on three key areas:

- **ease of integration**
- **enhanced application transformation**
- **performance and system management.**

The third of these areas includes the usual fine tuning to please the systems programmers and some improvements to the CICSPlex SM Web User Interface. But the most interesting developments come under the first two categories.

Integration

As IBM starts to put more meat on the bones of its 'on-demand' computing strategy, its prime integration objective has been to enable the re-use of CICS logic and processes within Web Services applications — by providing full access to standard APIs and communications protocols. This brings it one step closer to participating fully in those elusive Service Oriented Architectures (SOAs).

From IBM's perspective, delivering Web Services is all about support for XML and its associated middleware standards, WSDL and SOAP:

- **WSDL (Web Services Description Language) is an XML-formatted language used to describe a Web Service's capabilities — basically a definition of the service**
- **SOAP (Simple Open Access Protocol) is a lightweight protocol for exchange of information.**

These two standards have grown out of a joint development between Microsoft and IBM, and are widely deployed together. SOAP has been available as an add-on function to CICS for 18 months now, attracting some 800 users who are keen to use SOAP to help simplify the management of transactions and calls between CICS and the distributed world. In V3.1 SOAP becomes a standard feature, working alongside more heavyweight transport mechanisms such as WebSphere MQ (MQSeries).

The strongest message to come out of V3.1 is that CICS can act not only as a provider of services but as a requester too. This latter function is far less mature. One reason is that much of the IT industry's attention has, so far, been concentrated on unlocking the existing data and functionality already held within a CICS sub-system — in effect to facilitate distributed applications to tap into what exists.

The added requester capability opens up a new role for CICS — potentially allowing it to deploy Web Services in ways that will be of particular interest to strategic mainframe users. Among such users is Charles Schwab, long considered to be one of IBM's most influential mainframe customers, which was quoted at the announcement as saying: "*we need IBM to enable CICS as a service provider and eventually as a consumer, and look forward to the day when CICS is fully Web Services enabled.*"

To achieve this server/requester capability, CICS TS V3.1 includes a number of features for distributed transaction co-ordination, using WSDL to define the services. There is also a new tool, CICS Web Services Assistant — a

Figure 3.1: What is a Web Service?

- Definition: A Web Service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols.
- Web Services Description Language (WSDL) allows enterprise application services to be described in one standard form regardless of the hosting platform – WebSphere, CICS, .Net,....
  - CICS hosted services can be published alongside Web Services hosted on any other platform
  - WSDL accommodates optimizations – for example, binding other than SOAP/HTTP

build-time capability provided to create a WSDL document from a simple language structure or a language structure from an existing WSDL document (with support for COBOL, C/C++ and PL/I).

Another enhancement to integration in V3.1 is improved HTTP support. Significantly, the new CICS version is compliant with HTTP 1.1, with outbound support added — once again confirming the role of CICS as both server and requester (Figures 3.2 and 3.3).

Additional new Web Service capabilities for CICS include:

- **persistent sessions (as a default for interactions between CICS and a remote partner)**
- **support for pipelining and chunking of messages.**

The third key component of CICS' Web Service integration is security. As well as support for Secure Sockets Layer (SSL) 3.0 over IP, the new technology offers a range of key functions which will enable closer control of security within a Web Services environment. These include:

- **AES cipher suites with 128/256-bit encryption**
- **certificate revocation lists**
- **closer control of session IDs across a mixed sysplex**
- **support for mixed-case passwords.**

There is a strong sense here that IBM is trying to address many of the security management issues that can cause an obstacle when building bridges between CICS and distributed Web Services. These are significant steps forward.

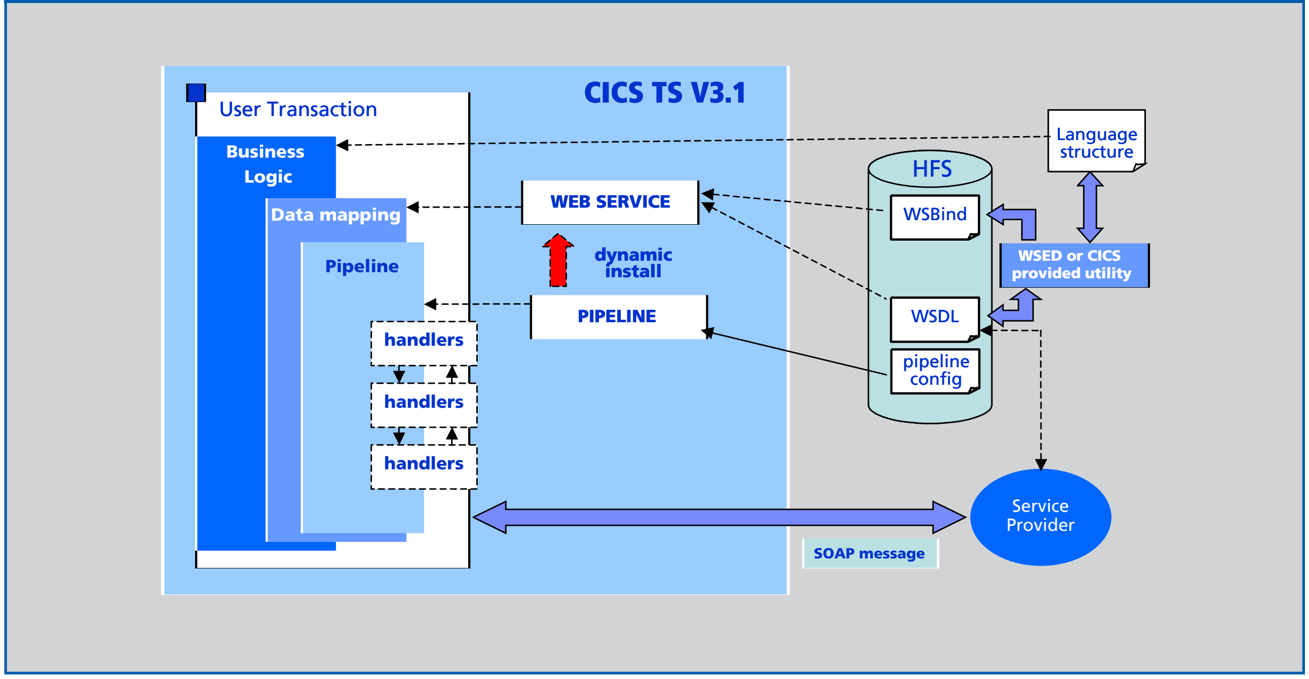
Application transformation

The other set of announcements for CICS TS V3.1 focuses on development tools. These seek to remove some of the programming limitations that have traditionally hampered the integration of CICS into a Web Services environment.

For example, the enhancements to C/C++ support bring their performance up to the same level as applications written in COBOL, PL/I, and Assembler. WebSphere Enterprise Studio Developer has gained higher prominence as a universal development solution, while the Open Transaction Environment has been extended with support for COBOL, PL/I, Assembler and non-XPLink C/C++ OPENAPI application programs.

But the main development thrust in V3.1 is tackling the verboseness of XML, which is becoming a real problem for those CICS applications attempting to process parameter data in XML and other memory-hungry formats. While the optimized data formats that have traditionally been handled by CICS could easily cope with the 32KB COMMAREA limit, XML carries far more baggage.

Figure 3.2: CICS as a Web Services Requestor (Source: IBM)



IBM has consequently introduced the concept of containers and channels. These can handle inter-program data transfer much more flexibly than through a COMMAREA. CICS now provides EXEC API verbs to create and access these new transfer mechanisms.

All singing, all dancing?

All of these boil down to a major shake-up of CICS' peripheral functionality. This produces many more choices as to the way that host/Web solutions can be implemented in the future.

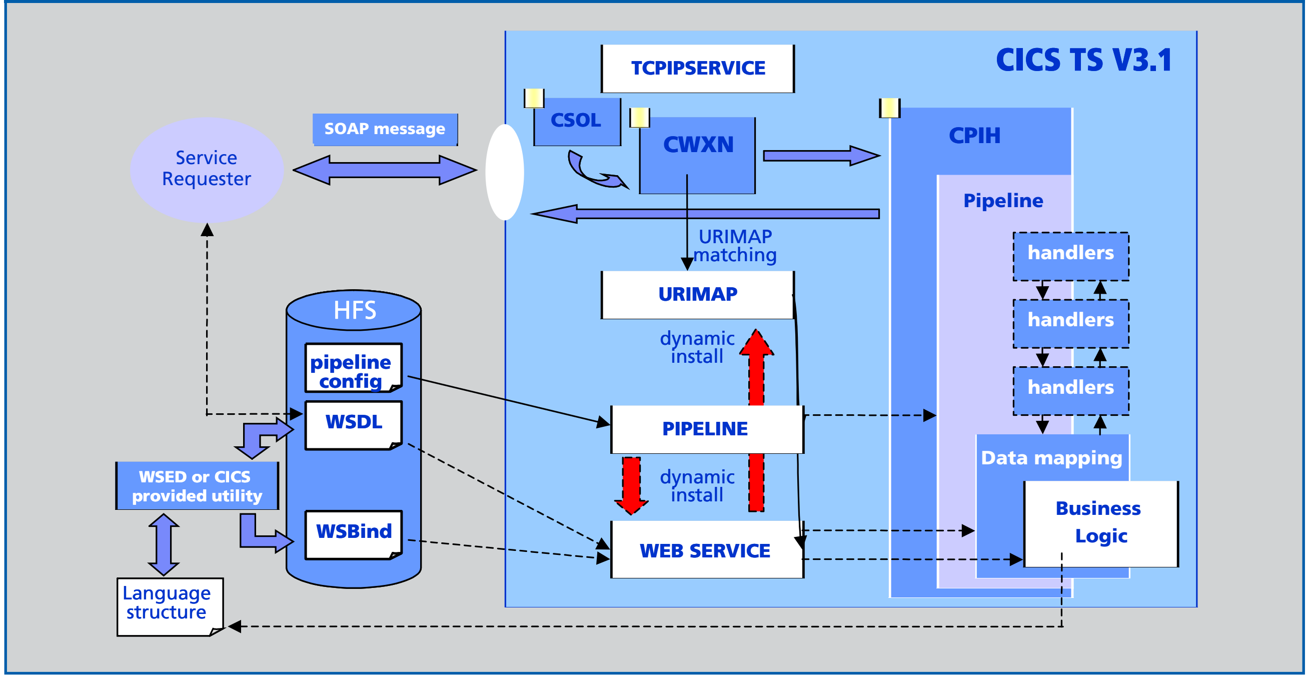
When it comes to opening up CICS applications to make their processes and data available to other platforms, there are already many options available to users. Even where IBM leaves gaps in its function set, these are still rapidly filled by independent software vendors — and CICS is no exception. For years companies such as Seagull, Attachmate, NetManage, NEON Systems and Clientsoft (the latter two having just merged) have offered services and products to customers attempting to integrate CICS applications with Web Services.

For some of these, V3.1 may be viewed as a significant threat. However, as Web Integration analyst Anura Guruge points out in his recent paper on the IT InDepth.com website, there is still a question of granularity here: the all-singing all-dancing solution that is being proposed by IBM might not be ideal in all cases. "*Until now, when it has come to CICS and Web Services much of the interest has been about representing specific transactions, or even sub-transactions, as separate Web Services. This is what Web-to-host host integration has been all about and from what I can see will continue to be all about. V3.1 does not, in my opinion, reduce the need for traditional host integration solutions such as NetManage's OnWeb 7.1 or even IBM's HATS v5 (though HATS still tends to lean more towards host publishing). Conventional host integration which relies on specific transaction capture, using the input/output fields displayed by a CICS application, provides the granularity to isolate and pick specific transactions from within a CICS application and then convert that to a Web Service.*"

In other words, V3.1 offers a whole range of new possibilities for CICS customers and, for the first time (within IBM's own product portfolio), this improvement really does enable a CICS application to be represented as a Web Service (Figure 3.3). However, in many cases, a more granular, transaction-focused approach to integration may be required; here there are a number of third-party solutions that can be employed.

Moreover, IBM itself points out that customers must make important choices between:

Figure 3.3: CICS as a Web Services Provider (Source: IBM)



- **traditional host CICS connectivity**
- **the kind of distributed Web functionality supported by V3.1.**

For developers in established MVS (z/OS) shops, a direct connection to CICS still offers superior quality of service and software maturity, as well as fewer APIs and system changes. The loosely-connected Web route into CICS is a better option for those looking to stay as open (as possible) to emerging Web standards, and who envisage frequent re-use of logic.

CICS as a peer player

What we see, then, is IBM laying the groundwork for CICS to act as a peer player in Web-oriented applications. With CICS enabled to function as server or requester, and full support provided for integration and security services within the Web environment, users will be able to take a considerably more flexible approach in re-purposing and extending CICS functions.

As IBM has said, the choice of Web-oriented or direct CICS connectivity depends on a number of factors — and these factors may change as

- **Web standards mature and mainframe support for them improves**
- **the availability, performance and cost characteristics and requirements for a given process or application change over time.**

The choice of a centralized host solution for new business logic may become more or less attractive depending on changes within an organization — mergers and acquisitions, changes in policy or cost and technical constraints. Certainly it is in IBM's interests to make sure that CICS (as well as other key mainframe sub-systems) remains as open and receptive as possible to new developments. But this will be achieved not by placing the mainframe back in the center of the IT architecture but by making it as flexible and as interoperable as other Web Service peers.

Of course, as mentioned earlier, there are significant numbers of legacy mainframe users around whose main priority is to migrate specific applications to different platforms with as little negative impact on the business as possible. They may face numerous problems in maintaining inter-program connectivity during the migration process —particularly if the move is incremental, and modules that make calls on one another need to keep track of location changes.

As Jody Hunt of Iona points out in his paper entitled 'Incremental application migration with Web Services' (Enterprise Data Center, January 2005) there are various issues that arise in such scenarios, including changes in data format, backup and recovery and security. Service oriented architectures offer a number of ways to support incremental change, by providing a flexible mechanism for inter-program communication. But for this to work effectively within a large enterprise, users need to be thinking about a much more sophisticated solution than those offered by vanilla WSDL implementations.

Strategic mainframe users want the ability to use CICS logic more flexibly within the corporate network, and that is what IBM is beginning to provide. Of course, this is only the first step, and the earlier quote from Charles Schwab suggests that it is expecting considerably more in future releases. In particular, IBM is likely to build on the WSDL support it has provided so far, by adding support for more protocols as well as focusing on switching and routing and location transparency.

Management conclusion

The signposts are now in place to show how CICS is now a credible peer player in Web Services, although this is a fast-moving field. There will certainly be more announcements to come. But, wherever users sit on the spectrum between strategic and legacy mainframe deployment, there is no doubt that recent CICS developments will remove many of the existing limitations to Web Service implementation.

IBM, of course, hopes that these improvements will generate new mainframe business among customers that are attracted by the declining cost of mainframe technology and the potential scalability and availability characteristics. One thing is for sure, though: the chasm that once separated the MVS-z/OS architecture from that of other platforms is closing fast.

Service management

Dr Keith Jones
IBM Software Solutions Worldwide

Management introduction

Many enterprises have now started to acquire Web Service skills and have implemented early Web Service projects. As these projects have become more numerous and more focused on core business functions there is a compelling motivation to invest in management solutions that monitor and manage services in line with business goals.

Many enterprises also understand the value of longer term strategies that match investment in Service Oriented Architectures (SOA) for the infrastructure to support applications and integration with greater business responsiveness and flexibility. Long term realization of an SOA is not the same however, as an initial investment in pilot Web Service projects. A critical element of that longer term view is a systems management capability that is based upon the correct layering and encapsulation which can deliver improved alignment between IT resources and business objects.

In this analysis Keith Jones reviews recent developments in service middleware and management technologies and looks forward to the capabilities that will deliver well-rounded SOA solutions. Success in achieving well managed service oriented systems architecture involves a fundamental change in approach to management — a move away from vertical stove-pipes and toward horizontal layers of re-usable resources.

Feeling the pulse

Management of IT systems has matured over the last 40 years and is by now one of the most sophisticated of all disciplines in enterprises, both large and small, around the world. During those 40 years, hardware and software technologies have evolved — producing an enormous variety of devices, connectivity options, processor types, storage hierarchies, batch processes, distributed computing processes, work flows, Web applications and many additional forms manageable IT resources. These are 'enterprise systems resources' (Figure 4.1).

These IT resources — often distributed widely across divisions, regions, and in some cases continents — are the fabric of enterprise systems today. Management of both hardware and software resources has become both a complex and sophisticated aspect of aligning IT resources with business goals for many enterprises.

That said, most management applications are often considered to be part of the overall middleware infrastructure — neither quite business application logic nor operating system logic. What distinguishes management applications from business applications is that they focus on IT resources, their properties and relationships — and not on the real world goods and services provided by an enterprise.

There is, however, one very significant connection. Whereas business applications model the real world operation of the enterprise at some level, management applications model, measure and manipulate the operation of the IT implementation for those business applications. Let me illustrate:

- **one of the functions of an order processing application is to create orders**
- **a corresponding order processing management application monitors the implementation to ensure that order creation functions correctly and efficiently.**

Management applications are fundamentally based upon a control feedback loop. There is a clean separation of responsibilities between resources, monitors and managers in this loop:

- **the monitors observe resource behavior and raise alerts when that behavior is outside a predicted range**
- **the managers react to alerts by adjusting provisioning for the resource and taking corrective actions to restore normal operations.**

Take the order processing example one step further. Orders are processed whilst an order processing monitor measures success and failure rates and order processing time. When orders arrive at an increasing rate more computing resources are consumed and the monitor issues an alert when order processing time exceeds an acceptable threshold. The order processing manager may react to such an alert by provisioning additional order processing server capacity.

As management applications become more sophisticated, so the metrics are refined, the boundaries and thresholds are normalized, the alerts are specialized and management reactions are fine tuned to business needs. The greater the level of sophistication in pre-coded management capability, the more 'autonomic' the infrastructure becomes — and the more self-managing IT operations become.

Most enterprises have complex heterogeneous application systems with mixed capabilities for management of those systems. With basic resource life cycle support, monitoring and measurement support and some level of problem detection and correction it is possible to feel the pulse of a business through the eyes and ears of its management applications.

Management by layers

Certainly one way to address management of IT resources is top-to-bottom and in vertical slices. This approach matches the needs of departments and divisions that act as cost centers or that wish to assert autonomy over a specific set of resources. Examples of this can be found in almost every enterprise. Purchasing departmental systems are often managed this way.

The problem with this vertical approach is there are often significant efficiencies to be had from pooling or 'virtualizing' resources horizontally as they are allocated dynamically in reaction to demand — and not statically as occurs when most departmental budgets are set. A horizontal approach to the management of IT resources — management by layers — would seem to be more logical, from many points of view.

The recent rise in popularity of Service Oriented Architecture for business systems has introduced a new focus on horizontal layering of IT resources (Figure 4.2). At the service layer, service interfaces are identified to encapsulate legacy and other application components as core business functions, and a mechanism is suggested for horizontal integration of systems that were once vertical stovepipes.

At the process layer, service interfaces are invoked as flows to handle critical business activities. An order processing flow might:

- **create new orders**
- **allocate or requisition inventory**
- **activate billing by invoking the appropriate services**

that are implemented (in turn) using legacy application programs and databases. At the business model layer it becomes possible to monitor and manage the landscape of flows that implement major business processes.

The introduction of service interfaces and composite process flows suggests a new way to manage IT resources, monitor business activity and measure effectiveness in horizontal layers. At the same time it also suggests that resources can be managed discretely to meet the needs of core business processes — in effect, bringing IT operations into closer alignment with business operations.

Some of the more sophisticated techniques now being applied at the lower (component) layers may be applied in time to the higher layers. For example, autonomic computing is now being introduced in IBM platforms at the lowest layers to manage automatically component failure detection and recovery.

Logically some of these same sophisticated management strategies will be applied to higher layers in time, giving rise to autonomic services and business processes. But for most enterprises, this vision remains several years in the future.

What has to be done now to bring the benefits forward? For most, the vertical management applications must evolve to:

- **recognize a service-based partitioning of IT resources and business activity**
- **provide new horizontal 'dashboard' controls for core business processes.**

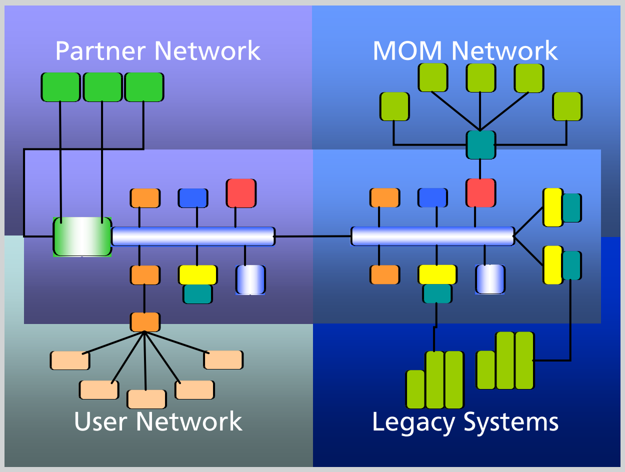
The service view

Services introduce new sets of IT resources. These must be managed together with existing legacy resources (Figure 4.3).

Even though many aspects of service software are being standardized, the artifacts actually created in deployed systems are not standard. Unfortunately a different set of implementation artifacts are likely to be found on each of the major vendor platforms that support services — such as:

- **service descriptions**

Figure 4.1: Enterprise System Resources



- **service registries**
- **service endpoints**
- **service gateways**
- **service implementations.**

Service interfaces are some of the key architectural components that must be carefully managed. Lifetime support for construction, discovery, evolutionary change and destruction of service interfaces in middleware infrastructures is critical to success. Management of versioning for service interface artifacts across deployment platforms is particularly critical to long term deployment of service systems.

In this picture, each service interface has a description that conforms to the W3C WSDL standard containing definitions of XML types, messages, operations, bindings and endpoints. Each of these artifacts becomes a manageable IT resource once the service is deployed in addition to the resources that are needed for a specific service implementation (Figure 4.3). In many cases a service interface is an encapsulating layer used to provide access to application components that are already conventionally managed resources.

Additional operations, new bindings, updated endpoint configurations and many other changes may be made during the lifetime of a given service. Management applications must not only recognize a service from its description but also understand all the artifacts, their properties and the relationships between them in order to monitor and control effectively a successful deployment.

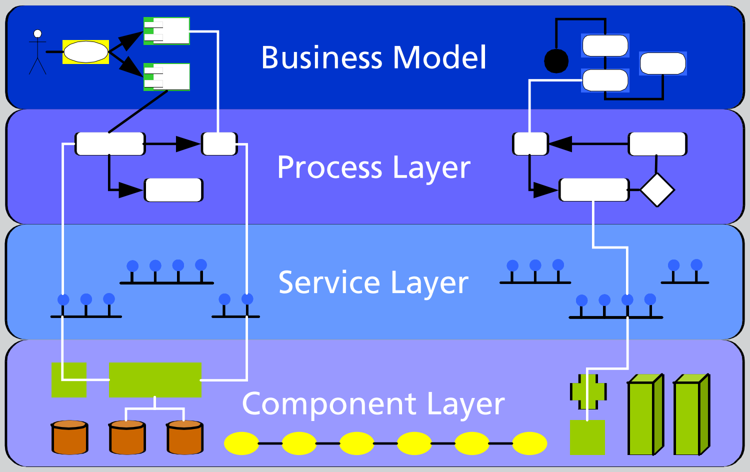
In a typical service architecture there may be many hundreds of deployed service interfaces with their implementations and an even larger number of service requesters (clients). The middleware infrastructure for such an SOA deployment provides secure and efficient access paths between requesters and providers.

In sophisticated scenarios a 'Service Bus' may be needed. This can provide implementation and location transparency as well as the bridge to legacy messaging systems and applications.

Management applications for services must monitor and manage the infrastructure carrying service traffic as well as the configuration of service providers and requesters. This infrastructure could be as simple as a SOAP/HTTP Web Service connection — or it could be arbitrarily complex as it spans departments, divisions, regions and national boundaries with built-in service registries, gateway connections to business partners and a multitude of sophisticated adapters for legacy integration.

Another service artifact that is critical to success is the service level agreement (SLA). Whilst implemented differently

Figure 4.2: A Service Oriented Layered Architecture



on every service platform. Such an agreement specifies the qualities of services offered by a provider and accepted by a requester. Management applications must be available to monitor and enforce operational compliance with service level agreements as they are deployed.

Such management applications will most often monitor services for availability, failure rates and performance characteristics. Other more sophisticated qualities of service may also be monitored and managed in time, such as security contracts and breaches (unauthorized access), integrity contracts and breaches (message replay) and transactional contracts and failures as they occur.

The combination of service providers, service requesters, service infrastructure and service management capabilities together comprise the future for service-oriented IT operations. But how will such operations mature in the coming years to provide the sophistication needed to support industry-leading and truly responsive business systems?

Collective wisdom

One of the many forces at work is a standardization project at OASIS that is focused on Web Services Distributed Management (WS-DM). This project is supported by most of the traditional management vendors (including Hewlett-Packard, Computer Associates, IBM and many others) and some newer vendors (for example, Amberpoint or Westbridge) which are focused on Web Services management.

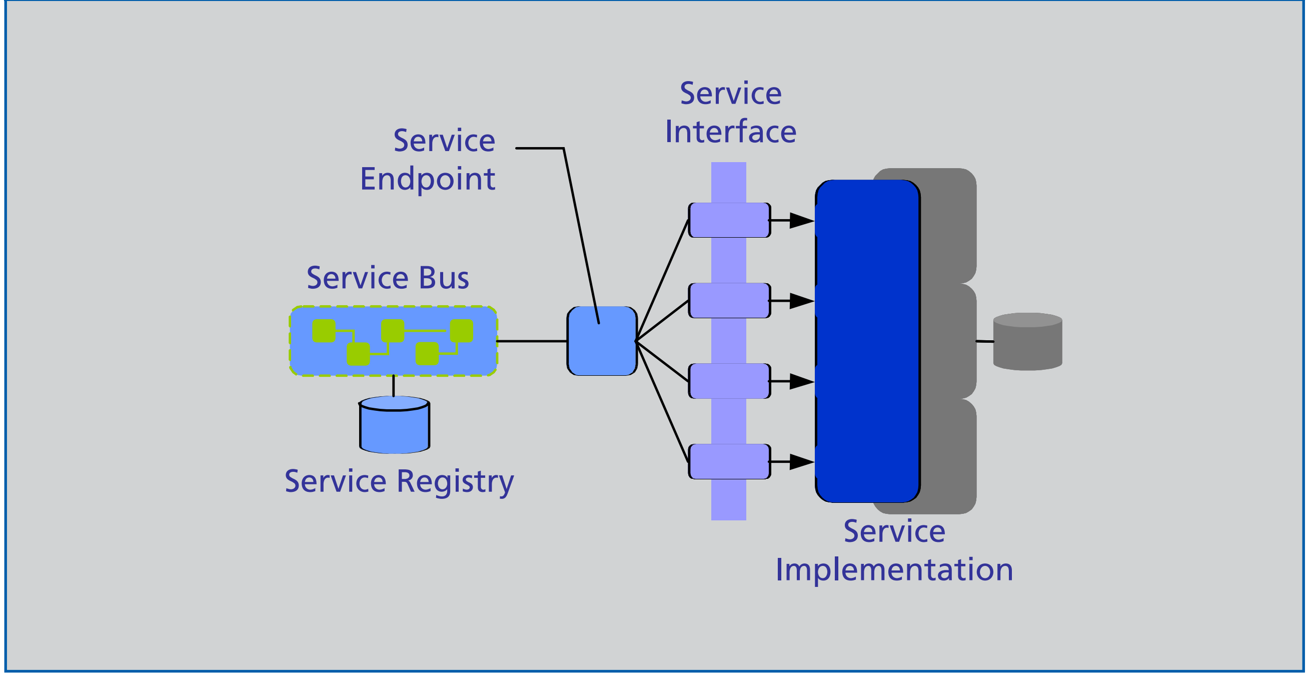
The far-reaching objective of this project is to define a standard way to integrate the management of all IT resources, using Web Services. The approach being developed would allow for resources of all types to be managed using service interfaces — even if those resources are also services in their own right.

WS-DM recognizes that all IT resources have properties that reflect current state and interfaces that accept commands to change that state during the normal course of an operational lifecycle. The service management architecture being developed (Figure 4.4) cleanly separates concerns between management applications, resource managers and resources in a manner similar to earlier management standards such as SNMP, WBEM and DMTF.

In this architecture the management application is a consumer of management information coming from one or more resource managers (called 'manageability endpoints'). Each resource manager has a service interface that complies with WS-DM specifications: this may be used to monitor and manage one or more resources.

Access to internal resource state is assumed to be possible using resource-specific APIs. Where a resource is actually

Figure 4.3: Service Artifacts



also a service it is possible, but not required, to compose the manageability interface with the functional service interface.

The development of this standard approach to management promises to deliver a migration path for existing management applications into the service era as well as provide design patterns for new applications that will focus on services and flows in the context of core business processes.

Unfortunately this new standard is not yet ready for widespread adoption. Realistically it will take some, or many, months fully to be developed and then obtain agreement.

Emerging solutions

Many enterprises have already started to define and implement their strategy for services. These have, by now, deployed into production only a relatively small number of Web Services. This factor alone has limited the need for, and therefore the size of, a marketplace for service management products.

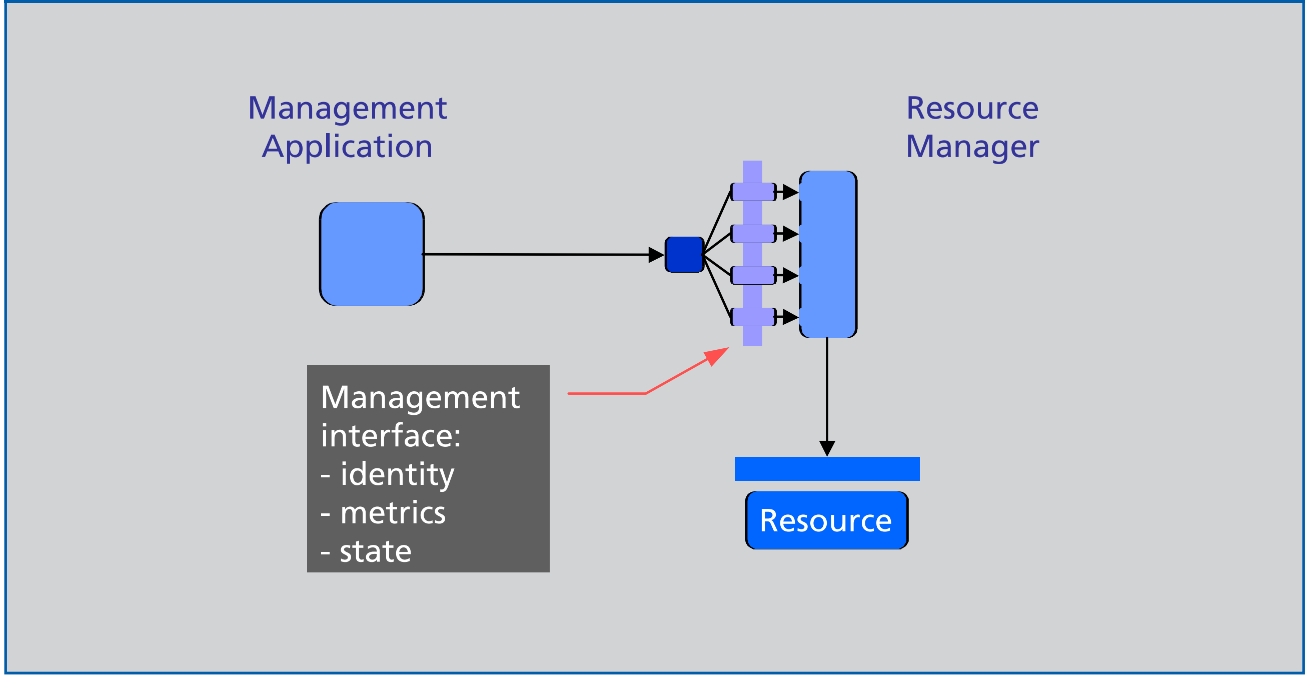
Another constraining factor continues to be the stark realization that deployed service artifacts differ from platform to platform. Thus different management products are needed for different vendor platforms.

Recent surveys have shown that the most popular service deployment platforms are already well established application server platforms like IBM's WebSphere Application Server (WAS), BEA's WebLogic and Microsoft's .NET — which have all evolved to become 'service-aware'. It should be no surprise, therefore, that the emerging service management solutions are already well established management products that are designed to monitor and control the artifacts deployed on those platforms.

As the development and deployment middleware platforms become more service-capable it is to be expected that they will include service management function. For example, WAS V6, made available at the end of 2004, includes basic autonomic capabilities and common event infrastructure that will assist plug-in J2EE management solutions. IBM's Tivoli Monitors, HP's OpenView, CA's WSDM and other established management products will undoubtedly soon evolve to become more service capable.

There are also many other vendor packages emerging with service capabilities in support of early adopters of SOA. Amberpoint, Westbridge, Blue Titan, CapeClear, Infravio and many others now offer solutions for selected aspects of service management for popular development and deployment platforms. As the marketplace for such solutions grows, the number of vendors offering management capabilities will also continue to grow.

Figure 4.4: Service Management Architecture



The long term trend that underlies the evolution of this service capability is the fast expanding use of meta-data to capture information which models application components as they are developed and later deployed into production (Figure 4.5). The use of meta-data to support inter-locking development and deployment lifecycles is an important step on the road to highly automated systems for the future that are based on semantics.

IBM's Rational, as well as many other vendors of development tools, are converging over time on a single development platform based on the Eclipse.org open standard. Eclipse platforms incorporate a plug-in framework and meta-data language that is easily translated into UML, XML or Java executable code.

The WAS deployment platforms are also to converge over time on a single set of middleware components that are meta-data driven. Long term we can expect to see automated management function in these platforms and sophisticated development tools based on meta-data that models services and their implementations as well as the infrastructure on which they run.

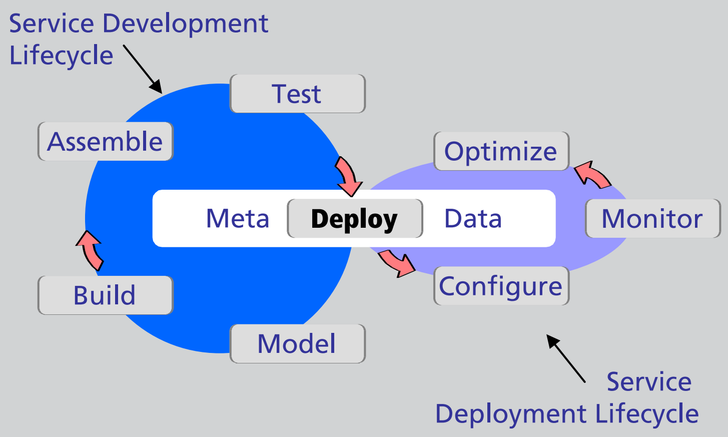
Management conclusion

Systems management applications have been developed to a high degree of sophistication over several decades and have embraced several fundamentally different technologies in their day — structured modular programming, client/server systems, message-oriented systems, distributed object oriented systems. Service-oriented systems will be no different. However, as more and more enterprises migrate to service oriented architectures there will be a significant opportunity to change approach to management of IT resources away from a vertical cost-center view to a more horizontal business process/competitive advantage oriented discipline.

Service interfaces for core business function have been proposed as the boundary of choice for identifying units of reuse for optimal time to market and the mechanism of choice for the integration of heterogeneous application systems for responsiveness in an 'on demand' world. A significant body of vendors and users are now aligned behind open standards initiatives to realize these benefits, whilst preserving the value of choice between competing solutions.

The development of standards for management at OASIS is a current and well supported initiative that aspires to deliver a migration path for established vendor management solutions as well as provide opportunities for vendors wishing to enter the service management marketplace with new offerings. The draft WS-Distributed Management standard is designed to exploit other proposed or already

Figure 4.5: Interlaced Service Lifecycles



agreed standards for Web Services — such as XML, SOAP, WSDL, WS-Resource Framework and WS-Addressing.

There are also emerging solutions for service management being made available by leading vendors — both established ones — such as IBM and HP — and new ones, such as Amberpoint and Westbridge. At the same time service infrastructure tools and middleware products are also emerging.

The future for these and related solutions looks well set to becoming more and more intelligent as they converge of a consistent set of semantics for services in the future.

Enterprise Service Bus — buzz term or business winner, techno-babble or critical enabler?

Nick Denning
Chief Technology Officer
Strategic Thought

Management introduction

Many business executives still do not believe that they receive good value for money from their IT expenditure. So much financial and management resource has been expended over the years — often implementing and re-implementing the same functionality in what appears to be the latest new technology fashion — that much credibility has been lost. Such a style of evolution seems to have added little and wasted much.

There have been two main drivers for embarking on such re-implementations in new technologies. The 'original' application may have become too expensive to maintain because it was not properly architected at the start and/or was developed piece-meal subsequently. Alternatively there may be a requirement to change technology that effectively mandates a complete re-write into the new technology.

As the associated risks are significant, it is important to find an approach that can enable a business to be supported in a more effective, manageable and timely manner. For some organizations the use of infrastructure may be appropriate.

In this analysis, Nick Denning considers many of the factors that need consideration if any infrastructure initiative — like an Enterprise Service Bus (ESB) — is going to succeed. As he observes, an ESB is a potentially useful technology but it will likely be irrelevant if there are not the organizational and architectural foundations in place to enable its exploitation.

He addresses the needs under the following broad headings:

- ***the case for software infrastructure***
- ***the ESB dimension***
- ***processes and road maps***
- ***supporting the business and quality culture***
- ***standards***
- ***the point of an architecture, and obstacles to introducing an architecture***
- ***architecture — top down and bottom up***
- ***no magic wand.***

The case for software infrastructure

In a previous analysis (MIDDLEWARESPECTRA, February 2004, page 18) I suggested that there is no 'one size fits all' infrastructure solution for all organizations, and that each must implement a solution appropriate to itself. I do not seek to try to offer 'a universal answer' here but by presenting a road map, I hope to identify the possible paths that a sophisticated business can take if it is to have a reasonable chance of success.

To achieve success it is necessary to focus on the factors that relate to the way in which sophisticated organizations design, purchase and deploy technology infrastructure. Their key aim is to obtain specific business advantage from the use of IT, and this applies to both users and vendors, for both need this type of software.

That said, many significant challenges face IT within businesses. For one, there is rarely sufficient time, or resource, to solve the 'whole problem' in any single project. This means that it is extremely difficult to take a process-based view and then implement that across a whole business.

Such an approach requires change to be implemented in ways that involve all departments. The consequent change management and technical rollout problems are frequently too large to be handled with on a single project.

The system designer, when reviewing the 'must, should and could' requirements, frequently only has time to implement the 'must' and a few of the 'should' requirements. Significant technology risk is also introduced if that system designer simultaneously introduces new technologies which are unfamiliar.

The common result has been that the typical approach used delivers individual stove pipe solutions (Figure 5.1) which each offer specific capabilities within a functional business area — and use a minimum number of technology layers (to contain the risks). Unfortunately, the resulting individual stove pipes can be both numerous, quite tall and use different software products. In addition, there is seldom the time to layer each stove pipe to facilitate re-use. Indeed the justification for re-use may be weak if such re-use of modules is not planned, especially if one or more subsequent projects use alternative technologies (or buy off the shelf solutions).

This reality has become increasingly unacceptable in large user organizations. It is no longer acceptable to possess a plethora of technologies and applications where duplication abounds or where business logic is continually re-written, often inconsistently, using different technologies and incurring high costs.

Neither is it acceptable to preside over an infrastructure that is fragile and brittle, where change is expensive and time consuming so that IT cannot respond appropriately to changing business requirements. Yet large businesses have to improve their ability to exploit IT to deliver competitive advantage.

If that describes the IT user, for an application provider (typically a software vendor) the problem is similar but subtly different. The application provider does not want to use too much third party technology — because the percentage of the price of the solution sold can end up being dominated by the infrastructure cost (this assumes that an end user organization has not already purchased the software infrastructure). When standard infrastructures are deployed in user customers, this situation will alter (but that is not yet the situation).

For example, bearing in mind that it takes 3-5 years to develop a successful application solution and establish it in the market place, it will be some time yet before even today's successful application vendors can expect customers to possess an infrastructure on which they (the vendors) can depend.

Thus, currently, software suppliers aim to minimize reliance on third party technologies. The result is that much infrastructure code is created, and re-created, by software vendors.

Thus both suppliers and end users contribute to the confusion in software infrastructure yet both can benefit from the adoption of standard infrastructures. This is an advantage that acceptance of a few (very few) forms of Enterprise Service Bus should provide, if the concept is adopted broadly enough.

The ESB dimension

That said, is an ESB just techno-babble or a critical enabler? Are we just discussing more buzz words or an approach that can be a business winner?

The problem I perceive is that there is a risk that many ESB initiatives will not deliver. This is because an ESB is not a product, though software products are required to implement one. Rather it is a design approach, even a concept or template which requires expertise and time to implement.

Before deciding to adopt an ESB, it is the view of Strategic Thought (based on our client experience) that each organization should determine:

- **whether there is a need for that organization to develop its own architecture**
- **the extent to which electronic integration is required between businesses participating in the value chain**
- **the level of legacy technology that is not J2EE compliant, and therefore requires integration with a non-Java range of protocols and technologies**
- **whether the size and complexity of the business warrants its IT infrastructure being considered as anything other than a single large — if multi-faceted — application.**
- **whether there are off the shelf products available to support the business which deliver an ESB and therefore mandate the organizational approach.**

For a small organization which cannot realistically be subdivided further there is probably no real requirement to consider an architecture involving an ESB. This size of organization would typically be a single business unit.

However most large organizations are made up of multiple business units, often using different technologies. These are connected by complex processes which, typically, interact with the processes of partners to create a value chain. In these circumstances the use of an ESB is likely to be vital — whether it is built in-house by that organization itself or is acquired from one or more IT suppliers.

With an Enterprise Service Bus, organizations should be aiming to:

- **reduce costs**
- **align investment to a business benefit**
- **demonstrate delivery against an RoI case**
- **empower the business to direct IT investment**
- **maximize flexibility to support change**
- **enable change more quickly than competitors**
- **manage complexity.**

The challenge that these organizations face lies in:

- **finding the technical and change management expertise that can enable such activities**
- **implementing a plan that is based on an agreed justification to upgrade the use of technology in a controlled and cost effective manner.**

Processes and road maps

Porter identified that "sustainable competitive advantage comes from building a value chain that is unique, dynamic and difficult to match". The most important objective of businesses is to identify, and then participate in, a value chain.

To achieve this there has to be an absolute focus on implementing processes within an organization that are based on standards. Furthermore, in an ideal world, such standards should also provide the basis for one organization's processes to interact with other processes in other organizations. The trouble is, if there are no lower level building blocks, implementing a process that can touch all aspects of a business is a huge task.

Delivering processes works effectively in short projects if there are underlying services that can be re-used. In this form, a top down business process focus is supported by a bottom up component implementation approach.

Short term process improvement must be underpinned by a long term road map that defines an architecture to support the business, a series of projects which will deliver to the business and against the time scales for these deliverables.

The difficulty is that the initial scale of activity — to put in place a coherent architecture — requires significant organizational change moving from teams aligned by functional stovepipes to alignment by process. Inevitably such changes are painful, and mistakes are made. Nevertheless, to me, this is an imperative for businesses that wish to interact efficiently.

Having accepted this, it is important to avoid the construction of processes that are either too large or brittle. They must be composed of sub-processes which can be re-used, and must be capable of being used in different ways without breaking.

Supporting the business

The challenge is to bring together people, technology and knowledge — and exploit these assets in a balanced manner. 'One size does not fit all' and we need separate teams with different value sets working to different business objectives yet still able to respect each others work.

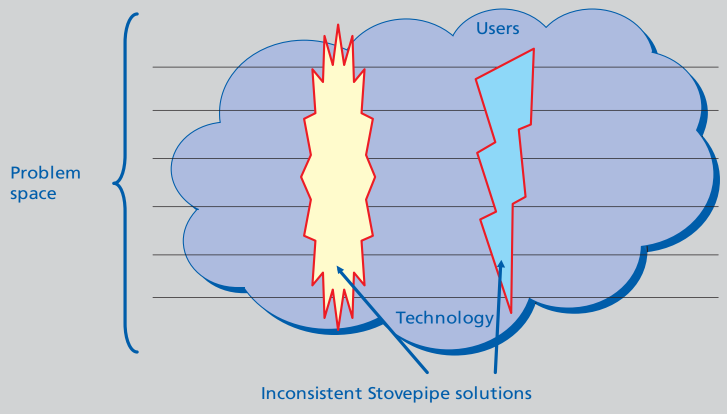
Yet organizations face dilemmas. For example many organizations observe the following on a regular basis:

- **central IT is seen as either being unresponsive or a cost burden that individual business units are not prepared to accept (or both)**
- **central IT is shut down, and the skilled individuals are deployed out to business units**
- **the business imperative for business unit managers is to support their own individual unit activities, so short cuts are taken in order to satisfy immediate business unit (not enterprise) objectives**
- **mission critical systems are developed to inappropriate quality standards — but they fail in a variety of ways**
- **the enterprise as a whole suffers, often with significant adverse consequences to operations (regulatory fines, lack of operational capability, inconsistency, large numbers of applications systems that do not communicate, etc.)**
- **business unit and/or enterprise executives are replaced**
- **new business executives arrive, IT is restored as a central function**
- **the merry go round starts again.**

This is a good example of 'one size does not fit all'. Elements of both approaches are required, but they must not grow 'fat' — they constantly need to sell to their customers and demonstrate value.

The real challenge is that, historically, businesses have organized themselves around functional silos of people doing similar tasks in the same place. Organizations now need to change their focus and provide management of business processes, with people collaborating in each process across previous organizational vertical silos. It also becomes possible, in such a scenario, to outsource those functions that

Figure 5.1: The problem with stovepipe solutions



are not critical to the key processes that represent the real value in the value chain.

Quality culture

Within this context, the 'one size does not fit all' still applies. This means it is necessary to use different approaches and attitudes to quality — and quality standards — appropriate to the task in hand. For example, the quality procedures that apply to building the software to control spacecraft or run nuclear power stations are different from those applicable when writing computer games.

Within a large organization there is, therefore, a constant need to balance the competing needs of time, cost and quality. One of the biggest challenges Strategic Thought finds is organizations adjusting to accept this.

Broadly, our solution is to have separate teams supporting each business function who have differing models of the trinity (of time, cost and quality). But this can only work if these teams possess mutual respect and co-operate when they need to interact.

Standards

Standardization is good because:

- **common products can be purchased from a minimum number of vendors; this reduces the costs of product purchase and maintenance, as well as introduces common skill-sets for staff**
- **when change is necessary there is a common baseline from which evolution can progress, thereby simplifying the process of change**
- **clear definitions can be provided to potential suppliers so that they can deploy into a recognized environment (or infrastructure); if an organization has adopted industry standards it is probable that suppliers will understand what exists which, in turn, will maximize the chances of successful introduction of new elements or software**
- **once a software infrastructure is in place it facilitates a reduction in cost of individual components, both existing and future ones.**

However the adoption of standards is not necessarily simple. It can cause challenges if:

- **inappropriate standards are selected, or if products do not exist which can be deployed or if the skilled staff to build, operate and work with standards are not available**
- **inappropriate standards are adopted, after taking into account the risks to the business (the danger is of adding excessive cost and time through trying to attain an inappropriate level of quality for the task involved)**
- **standards are implemented too quickly, as occurs if vendors have not yet implemented the selected standards and so lack products to deploy which exploit those selected standards**
- **standards are chosen but then fail to evolve (there are many examples of this, of which DCE and CORBA are only amongst the more recent)**
- **standards are introduced and made mandatory in an organization with minimal flexibility to deviate, even when a business case can be made for the use of products that are not fully compliant with those accepted standards**
- **over-engineering occurs, in order to comply with accepted standards.**

Architecture

Achieving the benefits from an architecture requires work. The following are the major activities we believe are required to develop an appropriate architecture:

- **the formation of a plan to identify the goals and objectives of the business, the scope for those goals to be supported and achieved through the use of IT and the development of a justification by which both business and IT management can determine when it is appropriate for the business in question to exploit IT**
- **the creation of a framework for the adoption and use by IT of standards to apply at any given time plus a plan for the development and extension of the formulated architecture over time as new technologies, standards and conforming products become available**
- **the establishment of a research and development function which can investigate new technologies in order to determine when it is may or may not be appropriate to adopt these within the said architecture**
- **the definition of all the IT assets of the business, and the initiation of projects to maintain those assets as the architecture develops.**

This represents a non-trivial effort. It must also be done in stages.

The architecture road map

An essential aspect of the construction of any architecture road map is:

- **identifying the capabilities that will be delivered to the business over time**
- **ascertaining the capital, and revenue, costs which will be incurred**
- **enabling the business to accept and operate the capabilities that are planned for delivery**
- **establishing, afterwards, the RoI finally achieved**
- **empowering the business to change its business priorities, and hence to adjust its business plans**
- **assisting all participants to accept the viability of the approach taken and, thereby, to understand the contribution required.**

Key to achieving these objectives is the coherent identification of:

- **architectural 'layers'**
- **the capabilities that will be provided in each layer**
- **the individuals who contribute to the definition of each layer.**

Typically this involves three primary layers (Figure 5.2):

- **business capabilities**
- **processes and applications**
- **infrastructure and technology.**

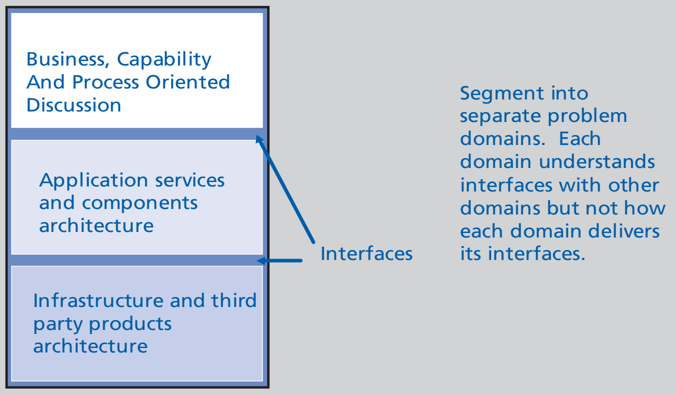
As relevant, however, are:

- **identification of the components of the architecture, the definition of the interfaces between each of those components and the identification of the people responsible for each of those components**
- **the inclusion of formal measurement mechanisms to enable the results of each activity can be reflected in subsequent architecture evolution.**

Within this, an overarching management function — as the design authority — is to:

- **undertake the requirements analysis, functional specification and high level design of all implementations and ongoing review of all projects in progress to ensure that the deliverables provided comply with the architecture**
- **ensure that the architecture remains coherent, and to take decisions where it is required to deviate from the architecture**
- **issue work to project teams, and acceptance of deliverables back from those project teams.**

Figure 5.2: Segmenting problems


Business, Capability And Process Oriented Discussion
Application services and components architecture
Infrastructure and third party products architecture
Interfaces
Segment into separate problem domains. Each domain understands interfaces with other domains but not how each domain delivers its interfaces.

It is also essential that the architecture team undertake the following long term functions:

- **IT and project risk management in support of the overall business risk management programme, specifically including contingency planning and risk mitigation activity**
- **identification and justification of corporate IT investments that cannot be justified by any one project**
- **change management to minimize the impact of IT change on business processes and maximize the ability to support business change**
- **interaction with HR to support training across the organization.**

Without these architecture functions it is typical to find some or all the following:

- **large amounts of poorly utilized hardware and software are purchased by individual projects**
- **mission critical systems run on unsupported hardware and software that incur excessive costs to maintain (with no plan for evolving into a supported environment)**
- **the risk of major business disruption rises, in the event of a failure**
- **the lack of software licence management results in high costs of unused software or under licensing of software that is used (which can have serious unplanned budgetary implications when discovered)**
- **the inability to leverage common components produces many separate and inconsistent stove pipe applications which have different data models, utilize different technologies, re-implement common functions inconsistently, lack intercommunication abilities and/or are difficult to engineer into a coherent business process**
- **no resource exists for analysis of the full impact of change; where the scope of a change is not understood, there tends have a major impact on the business when they are needed — the danger of 'rip and replace' is very real when the scope is not understood.**

Implicit in a formal architecture approach, therefore, is the evolution of that architecture over time by a series of small initiatives which minimize risk and provide benefits. Where no architecture exists, systems often evolve to a point where they become unacceptably fragile plus the cost of further modification becomes unacceptably high.

The difficulty is that there is often a temptation to scrap such a system and replace it with a completely new one — because the process of evolution is perceived to be too hard. Typically this decision underestimates the difficulty of replacement, particularly when two systems cannot be run in parallel and data must be migrated from the old system to the new. Generally it is only possible to do 'rip and replace' economically if the business accepts that it should change its business processes to support those that are provided by some new system (this is one of the 'prime attractions', for some, of SAP, PeopleSoft and others).

The Importance of the 80:20 rule

Pareto's 80:20 rule says that it is possible to deliver 80% of the capability for 20% of the cost. An organization must be able to prioritize requirements and determine in which phase particular requirements will be met, acknowledging that some will never be met.

One can allow some mitigation and contingency planning activity to minimize the risks of implementing requirements that it transpires must be supported. But the danger is that rearguard actions are taken by vested interests to re-introduce previous requirements, causing costs to escalate.

Phased implementation permits a review of requirements at each stage based on the 80:20 rule. This should be mapped against business priorities and is an essential aspect of the development of a cost effective approach.

Obstacles to introducing an architecture

With this weight of argument it would seem foolhardy — if not reckless — to fail to introduce an architecture along with the architects to manage and develop it. Regretfully there are some significant practical obstacles to the implementation of most architectures:

- **adopting an architecture requires significant organizational change; this will affect the present incumbents of the current structure and will (almost inevitably) encounter resistance (which can only be overcome from the top)**
- **to be effective any architecture team must be the primary point of contact with the business, define the architecture, commission projects and act as the one design authority; to ensure success, highly capable management and technical skills are required**
- **any architecture must evolve using lessons learned and be the result of a long term**

commitment which involves many stages as well as the opportunity to correct errors; the danger can be that the setting up of an architecture team sets unrealistic expectations of what can be achieved (and shelving of the initiative as a failure if these unrealistic expectations are not satisfied)

- **roles and responsibilities of many staff will change as will the criteria by which staff are assessed; training is, therefore, required in communication and management skills to enable staff to adapt to new roles**
- **significant up-front costs exist; an appropriate level of analysis is required to understand the overall cost structure of the business, and mechanisms must be put in place to measure the costs of the architecture, and the resulting business benefits generated (otherwise the only view of this architecture activity will be a cost)**
- **the absolute imperative for an architecture team must be a focus on delivery: there is a significant risk that an architecture team can create an ivory tower and lose credibility; thus an architecture team which is not responsible for implementing its architecture, which only performs an advisory function, is likely to fail.**

Related to the last point is another aspect. There must be a single 'command' structure responsible for delivery within the business for which the architecture is being developed. Fundamentally this means a single chief architect should be responsible for a business unit (or enterprise, if this work is done at the whole organization level).

Strategic Thought has seen scenarios where:

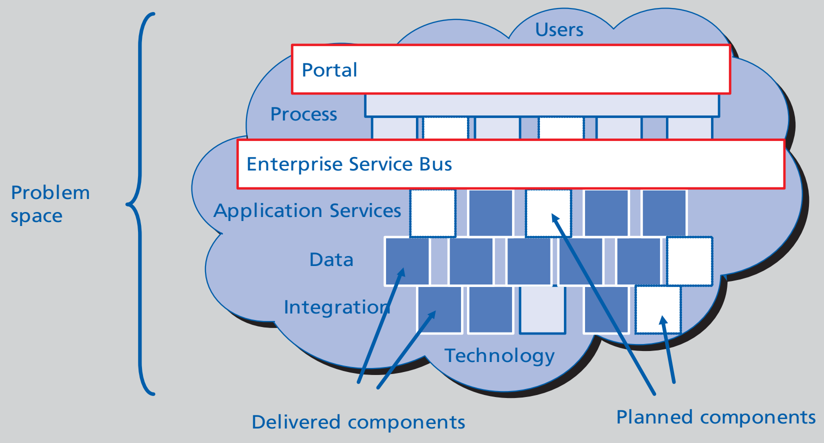
- **multiple architects 'in-fight' amongst themselves**
- **no lead architect exists — such as in a subsidiary where there is a general acceptance that it will follow the lead of the parent — but no specific direction from the parent is forthcoming.**

What should be clear from this is that the obstacles to a successful implementation of an architecture, and the benefits that can accrue, are people issues rather than technology ones. There is no point even in considering the introduction of such an architecture (Figure 5.3) if an organization lacks the commitment.

Delivering and using an architecture

The development and use of an architecture revolves around the following principles:

Figure 5.3: Architecture



- **there is an architecture team responsible to the business for the delivery of IT capability; the team will develop the architecture to support these requirements over the expected lifetime of the business**
- **an operational team will manage the production systems on a day to day basis**
- **development teams will be available to implement the projects defined by the architecture team; these should be viewed as 'sub-contractors' by the architecture team**
- **the architecture will be decomposed into a business view, an application/process view and an infrastructure view.**
- **development must not be a capital activity but must be treated as revenue expenditure; specifically this means that there must be no 'large projects', even with a well defined end point — instead it is vital that all large programs are funded as a series of many small projects**
- **the execution of many small projects means that expenditure must be considered as a 'rate of burn activity', with each project funded based on its own justification, and it must deliver within that to justify further investment in subsequent projects**
- **standards must evolve regularly, supported by the necessary level of training and development**
- **these standards must ensure that there is scope to change suppliers within a defined period of time (if ever required)**
- **whenever off the shelf software products are available they should be used in preference to custom creation**
- **vendors must be actively managed, and encouraged to develop and support components that will comply with the current as well as future architecture iterations**
- **the architecture framework will comprise a business, an application and a technology layer**
- **this framework will be decomposed into components with well defined interfaces; the detailed design and implementation of specific components can then be delegated to individual teams who can then focus on their specific delivery with minimal input from other project teams**
- **integration and acceptance testing will be a critical aspect of accepting deliverables from individual project teams and deploying those components into production**
- **the organizational structure will be designed to support the ability of that organization to outsource development in a controlled manner (though additional contract/project managers may be required to manage this approach).**

The objectives of defining a component are to:

- **enable individual components themselves to utilize lower level components, thereby minimizing the need to build large components from scratch and facilitating re-use**
- **ensure any component should be extendable, expandable, upgradeable or otherwise be capable of improvement within the scope of the overall architecture with minimal and well defined impact on the any of the components which utilize it**
- **ensure that any component which does not fully comply with the architecture is able to be included within the architecture, albeit via a documented risk and by wrapping such a component behind an interface that is consistent with the architecture**
- **facilitate the acceptance of changes which have an impact on the business processes**
- **maximize the flexibility of the business to define and develop new business processes by re-engineering its existing sub-processes.**

Architecture — top down dimension

Any architecture includes the need for top down analysis. The principles related to architecture identified above must be addressed during this analysis.

To deliver against these, an architecture team must:

- **understand the current business processes and how they are currently implemented**
- **determine what new business processes are required to support business change in the future, the time scales in which they need to be made available, the estimated costs for delivery, the development of a joint plan supported by a financial justification, the identification of priorities for implementation in the foreseeable future plus the development of a plan for the immediate period**
- **decompose the end to end business processes into sub-processes which can provide the components to re-engineer and evolve existing business processes — by re-organizing and**

supplementing these sub processes while avoiding or minimizing the change necessary within the individual sub-processes themselves
- **agree with the business the mechanisms by which new requirements will be identified and delivered, with appropriate involvement from the business**
- **share ownership of the process and logical data models with the business**
- **identify the delivery channels required by the business and seek to optimize the efficiency of these data models, working with the infrastructure team to select the appropriate infrastructure relevant to the various channels**
- **be responsible for risk management of IT in conjunction with overall business risk manager.**

Architecture — bottom up (infrastructure)

Infrastructure is the bottom up function. It is involves all aspects of the architecture related to understanding:

- **the current technology baseline**
- **industry standards**
- **relationships with suppliers**
- **provision of a technology road map for the evolution of the technology stack over time to meet the needs of the business.**

The infrastructure team needs to ensure that the business can continue to exploit technologies (particularly existing ones) that are not compliant with the new architecture standards. Nevertheless, its principle focus should be on:

- **describing and owning the technology baseline — all the IT assets that are contained within the business**
- **defining the architecture, including all the products, standards, methods and so on**
- **looking forwards at new technologies and assessing the implications of new technologies on the existing architecture**
- **determining if and when a new (or old) technology is relevant to the business**
- **considering the impact on the current architecture of upgrading the architecture and what needs to happen to applications currently operating**
- **evolving current standards, to minimize the impact on current developments of the expected new architectures**
- **taking responsibility for risk management to ensure the appropriate compromise between capability and value for money is achieved and to practise risk mitigation activities as well as adopt contingency plans**
- **owning — and continuing to update — standards, methods and guidelines associated with the architecture, including the training associated with the adoption of the infrastructure**
- **directing infrastructure suppliers.**

The application function

The application function is required to be responsible for all aspects of the architecture related to the delivery of sub-processes to the business. This team is therefore responsible for all aspects of the architecture that relate to implementation of business logic, which includes:

- **mapping of the logical to physical data model, as it is represented within each of the application repositories**
- **defining the transformation rules, to map data between physical models when data is exchanged between systems as processes flow across individual systems**
- **abstracting business processes that run across applications from those that operate within specific applications**
- **understanding of the business transactions that take place**
- **defining the messages that flow within each work flow — to enable processes to cross system boundaries**
- **identifying of the individual components of business logic that are executed within the individual systems and the interfaces that are available to each one**
- **understanding the functions provided within each of these application components**
- **introducing mechanisms to manage event and error processing, to ensure that business transactions either complete (and cannot be lost) or that errors are identified for subsequent correction or are canceled (or rolled-back) if it is not possible to complete them from a business perspective**
- **adopting mechanisms for providing management information on all transactions back to the business.**

It is critical that such an applications architecture is structured to identify:

- **work flows that represent a sub-flow which is unlikely to change once delivered**
- **super flows (those that use the sub flows) but which can be re-engineered or supplemented with new super-flows that exploit underlying sub-flows to deliver new capabilities**
- **business processes that need to be delivered today as well as changes to business processes that may occur over time — thereby enabling the design of the sub-flows to take into account possible future process requirements as well as minimize the need for changing sub-flows at later dates**
- **where user interactions are required at the boundaries of sub flows and thereby ensuring that such user interaction can fix any mis-match between sub-flows, thus enabling super-flows to be constructed from sub-flows without the need for code changes**
- **the use of people to manage possible mis-matches between sub flows with the needs of straight through processing (STP), which may be difficult to modify subsequently without code change.**

No magic wand

Make no mistake: an ESB — which can be regarded as being the latest attempt to deliver a rationalized software infrastructure — is not a magic wand. With an ESB it is still possible to waste large amounts of time developing unnecessary complexity that does not satisfy its original financial justification.

In the near future ESB initiatives therefore should only be undertaken by organizations which:

- **can identify substantial gains from participating in a value chain**
- **expect the competition to use such processes as a means to establish competitive advantage**
- **possess the necessary management resources — and commitment — to support the change process.**
- **have other compelling reasons for change — such as the requirement to outsource — which make major business and technology change inevitable.**

The good news

The good news is that, once there are established patterns for the development of an architecture embracing processes supported by an ESB — and shared by all vendors supplying products to create an ESB — then it will be appropriate for less sophisticated organizations to consider adoption. The amount of customization will fall and commonality will increase.

That said, implementing an architecture to deliver the benefits discussed will likely require giving consideration to:

- **the adoption of a portal technology which provides a common interface, encapsulates all underlying process and service logic and removes the requirement to re-write applications only to provide a common look and feel**
- **the introduction of a reliable infrastructure based on ESB patterns that will provide indirection to underlying services, support Web Services standards (where appropriate), integrate with legacy protocols where required and provide interconnectivity within an organization and connection to its partners**
- **bottom up provisioning of services that comply with Web Services standards, by utilizing new software and wrapping legacy services that will be used in the future**
- **defining a business justification for all work, as well as basing this on a revenue model (rather than a capital one), by implementing a number of small projects each delivering against an expected RoI and by adopting a model which allows for progressive development and evolution of capability**
- **provision of sufficient commitment to R&D to ensure that a coherent architecture evolves**
- **producing the necessary business management resources to ensure that suppliers and individual project teams deliver to the requirements of each project, and support the long term architecture as it is evolves**
- **accepting major changes to the organization's structure and way of doing business so that this aligns with the architecture (this may mean defining terms of reference for each group to enable them to operate autonomously to deliver the required components).**

Management conclusion

As Mr. Denning indicates, those who 'hope' that they can introduce an ESB overnight are unlikely to be successful. A software vendor cannot supply an ESB. An ESB is a technology enabler for business change. Each organization must *first be sure that it has the appetite for, and capability to deliver, change before investing in more technology.*

An ESB, as a manifestation of infrastructure, requires steady progress to be successful. It needs to happen in conjunction with business partners and suppliers as well as one’s own colleagues within an organization. It will therefore be a challenge, particularly where activities are already in play to implement an ESB to support outsourcing plans to which the business has already committed.

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]