MIDDLEWARESPECTRA

incorporating FINANCIAL MIDDLEWARESPECTRA

Contents

August 2003

Volume 17 Report 3

Grids and Web Services make for a changing middleware environment

Anonymous

Management introduction

This discussion comes from a senior IT Executive in an international financial institution who wishes to remain anonymous. While it is not generally the editorial policy of the **MIDDLEWARESPECTRA** *to accept anonymous case studies, this executive has so much to say of value that we have decide to proceed.*

Part of our reasoning is that this executive sits in an interesting position with at least three quite different sets of responsibilities, including:

- ***infrastructure architecture***
- ***IT governance (including ensuring that projects observe corporate standards — in all senses)***
- ***delivery of a strategy for this Institution to harness and exploit new technologies that are imminent.***

This discussion addresses the evolution of middleware over the past 5-8 years and how it is likely to progress in the future. It examines where Web Services and Grid middleware technologies are likely to fit in the future as well as describing how and why IT is going to have to work in a different environment from that to which it has become accustomed over the past 20 years.

Big changes have happened

Five years ago this Institution was in a quite different situation to where it is now. It was a relatively early player in adopting Internet and Web based technologies. IT's focus, in those heady and transaction volume driven times, was helping its engineers to deliver scale. The need was to keep available an infrastructure that could cope with what appeared to be ever rising Internet activities.

That has changed. Since the implosion of stock markets, particularly those concerning high- tech stocks, this Institution has had to change. For example, Internet-related activity has reduced significantly.

Three years ago the Institution took a step back to consider what its business would look like in the future. Although it had succeeded in delivering financial services over the Internet — and had delivered solutions with technologies that were probably never designed to do what they were used for — it needed to reconsider where its business was going. It came to the conclusion that it had to become more agile than it was already.

Put another way this Institution decided it needed to provide IT resources that its business could leverage to accommodate the way that it (and its customers) wanted to do business — rather than having them constrained by IT. One of the dimensions this Institution noticed was the challenge presented when buying, or divesting, businesses. When that happens, no organization can avoid bringing in and integrating (or disposing of) some infrastructure. In an acquisition, this infrastructure would never have been a part of the acquirer's architecture previously.

What mattered here was that, if the Institution could not accommodate acquisitions, or make divestitures straightforward, it would hinder its ability to do business. The value of an acquisition comes when the separate parts can work as a cohesive whole.

The reality was that IT was not yet suitably cohesive. What was needed was an infrastructure where the Institution could implement plug and play on an enterprise and business basis. That led to the development of what was then a revolutionary idea of a truly open, service oriented architecture (SOA).

At the same time it was noticed that just coalescing at the service level does not necessarily move anyone any further forwards than just being able to integrate a smorgasbord of services. That is fine, but only to a point. But, if a business relies on having a cohesive picture of the end user or customer — for example — then this requires a different level of integration. Integration must happen at a data layer so that:

- **everyone in the Institution can talk about the same customer in the same way**
- **the customer sees that there is a deep, valuable relationship that is being built over time (this is what Michael Porter referred to as 'customer intimacy'.)**

That delivered a second insight — the concept of a virtual data store which lead to a year's worth of architectural investigation, particularly considering what:

- **constitutes an SOA (middleware) bus**
- **a virtual data store should look like.**

The conclusion was that a middleware bus was a lot further along in terms of actually being able to deliver production results in ways that were already recognized. In contrast, the closest anyone could get to a virtual data store (this was a couple of years back) was to build something like a physical data store for general purpose access. But this did not go as far as what was envisioned, so it was put on the back burner.

Having adopted the principles of a Services Oriented Architecture (SOA), the next development that arrived was Web Services (which, it could be argued, is the logical implementation of the SOA concept — as long as you remember that SOAP/HTTP does not equal Web Services). What was also relevant was that a major piece of the infrastructure already existed and was in use — namely a piece of middleware that enabled basically any distributed client to talk to the principle source of services for customers, the back end hosts. This implemented a request/reply model. It was expensive to run — but had the positive benefit of working. The task in building a request/reply bus is to make it open and standard and avoid dependencies on any physical tier.

Three styles of application interaction

At this point one of the Institution's infrastructure architects contributed some 'deep thinking' about how many styles there are of interaction between applications. The conclusion was that there are three important ones:

- **request/reply**
- **publish and subscribe (or pub/sub)**
- **'modernized FTP/batch' — the mass or bulk transfer of data.**

It was decided that if these three existed they would cover the vast majority of needs. As explained earlier, the Institution already had request/reply working. It went and did a lot of work on pub/sub — because it seemed to have the most immediate requirements. The need for 'Modernized FTP/batch' (MFTP/b) followed behind.

Request/reply and pub/sub

The first point to make is that the difference between request/reply and (say) pub/sub is often more a stylistic issue than one about function. While there are differences, the real issue is how does an application see itself and what is it trying to achieve.

That said, the principal difference is this. In a pub/sub infrastructure, the publisher of events is unconcerned with what happens to those events. It is like the radio: the programs are broadcast. If anybody cares to listen that is great. If only a few, or nobody, cares to listen that does not affect the publisher (at least, not in the first instance).

Most organizations, whether they realize it or not, could have uses for the pub/sub model. When business objects change state, publishing that state change can be useful. The advantage is that other applications do not have to poll in order to find out when a state change has occurred. Yet, once the state change is received, by any number of subscribing applications, this can trigger multiple subsequent actions.

Even better, unlike request/reply, pub/sub does not require that you predict who or what is going to need the state change information. This means that different business units, for example, can add themselves (and remove themselves) as subscribers. All they need to know is what information is being published and how to read it.

Going back to what was said earlier, this makes acquiring and divesting of businesses that much easier. Less has to be changed or altered in the infrastructure.

The contrast with request/reply is that the latter means that the requestor cares deeply about the fact that there is a service which will perform a function upon request. While one can try to minimize whether or which provider knows, this can only go so far. From a business point of view, it has the disadvantage of being more tightly coupled, so the greater determinism comes at a price. In summary the requesting 'clients' care that something has to be done — a response to an enquiry received, a transaction processed or whatever. They want results or some action to take place; they want to know that somebody did what was asked.

There is, therefore, a big practical distinction between pub/sub and request/reply. It is about who needs to know what happened when, and how. It has a great deal to do with service. Pub/sub offers an extra degree of freedom.

Having said that, remember that it is always possible to mix and match. A pub/sub event can kick off a request/reply service which might publish its result. Equally a request/reply service might exploit pub/sub to obtain information that is being broadcast — say about bond or FX prices. It is perfectly possible to hook each up in different combinations.

The knack lies in choosing the right one for the right task, for it is quite possible to choose the wrong model. Indeed, it is a fact of life that most developers and even architects have been brought up in a request/reply world. They 'naturally' think of using it when pub/sub might be more appropriate. For example, it is possible to create pub/sub in the request/reply model by using call back; but it is nowhere near as efficient as 'pure pub/sub'. The key is to encourage developers to think about how actions and events occur and then decide which model is more applicable.

MFTP/b

In 'MFTP/b' the coupling is even tighter still, in the sense that what the connections will be are pretty well understood in advance. In this it shares characteristics with request/reply.

On the other hand, a MFTP/b 'writer' is a publisher of large amounts of data (akin to a batch in some senses). The prime difference is that you usually create a datastream because you know what is going to be needed next. This will have, in effect, one publisher (or creator or writer or whatever), with only a very small number of well defined, in advance, readers of each MFTP/e datastream.

In practical terms, attention has only recently focused on the architecture for MFTP/b. The architecture is only now from high level design to implementation in order to create a 'MFTP/b bus' — to work alongside the pub/sub and request/response middleware buses.

Implementations

Already the pub/sub bus implementation has been operational for over a year. It supports JMS/Java and COBOL applications, but can be extended as needed. The interesting thing is that it is proving to be quite a success. In the beginning the problem was trying to convince people to use it rather than build their own examples of point to

point messaging. It took a long time to obtain results, partly because people were initially trying to work out how they could justify the avoidance of its use.

But, after some 18 months, people are noticing that all sorts of events are being published which they can use. Now there is the challenge of worrying about those who are using the pub/sub bus too aggressively (where they are really publishing something that might be better suited to, for example, MFTP/b). It is also popular because it is a common infrastructure resource for which business units do not have directly to pay.

In the request/reply case, work continues on building the new bus, using Web Services to service C, Java (JAX-RPC) clients and COBOL services initially. It is currently in engineering. The plan is to have this working in 2003. One of the targets is seamlessly to migrate all current users of the current proprietary request/response middleware to the new bus — by using a thin abstraction layer on top of the new open standard request/reply bus.

The savings have the potential to be considerable. The expectation is that the Institution will be able to retire a whole physical tier of infrastructure that is currently dedicated to coupling together distributed clients to the various different forms of host. With this new bus an intermediate tier can go.

In addition, advantage is being taken of the new facility in COBOL to support XML. The middleware bus is, in effect, neutral. Clients, and servers, put their requests or responses or events or streams into a neutral format that makes sense to them. This is then conveyed on the appropriate bus and rendered into an understandable form when it arrives at the destination — which delivers huge flexibility.

Web Services and Grids arrive

If Web Services was one new arrival, a second is the Grid. The common key here is that all middleware is now being written to conform to accepted open standards. Thin wrappers will only be used where absolutely necessary.

Now the focus is to move to using other implementations of standard in order to:

- **concentrate less on the middleware and more on the business**
- **obtain greater platform coverage**
- **deliver at a lesser cost (than before)**
- **reduce maintenance overheads.**

Web Services contribute to this. They are based around standards. This is where the Grid has its place — because it is also becoming standardized.

Involvement with the Grid occurred because one of the infrastructure architects went to a conference and was talking about architectures and somebody asked whether he had heard about the Grid. This was a couple of years ago and much of what was being enthused about then did not seem real at that time, although some research did occur which suggested that:

- **'this Grid stuff would be great only if it can be married to standard Web Services'**
- **it would work best on applications that were compute intensive, but this was difficult to relate to the finance sector (much of Grid computing was then dominated by 'big science', compute-intensive types of applications).**

One thing that changed many people's thinking has been the way that IBM, and others, have adopted the Grid and actively promoted the adoption of standards. This was like a first domino falling.

The second domino was the Institution changing its business model. Without going into detail, this represented a major change because we have to be able to provide future services in ways that are affordable and acceptable to customers.

When you start thinking about delivering this it does not take a rocket scientist to appreciate that this involves being able to look at each customer's accounts and any associated investment portfolio. In turn this requires serious computations if it is to be meaningful, especially if risk analysis dimensions are included.

The basic financial issues are well understood here. There are practical solutions available within the finance sector. Nearly all are compute-intensive when done well. The industry challenge is to deliver them in ways that are affordable.

In this context, Grid computing now made sense. By exploiting Grid-type capabilities (to use spare computing cycles that would otherwise go unused) to perform the analyses, it would be possible to:

- **use what we already possessed to provide the raw computing power**
- **deliver analyzed information to branches and customer facing staff**

- **improve the interact quality with customers.**

A proof of concept project has worked well. The next step is to move to a pilot production project. So Grids are making real progress — because they are based on standardized concepts and middleware (including Web Services).

Lessons learned

It can be argued that the whole IT industry — and few financial institutions proved immune — lost sight of the goal for a few years in the late 1990s. When there was so much money running around, aided by venture capital, attention to what mattered was lost. Unfortunately, for many people, this has proved to be very painful because any model that does not care how effectively things are done is unlikely to last. And so it has proved.

The point is that too many became excited about the technology and too few thought about how the investment was actually going to improve the underlying business. Too much was funded on the basis that it sounded great. In practice, as most have learned over the past three years, the difficult, hard, heavy lifting work takes considerably longer than the act of 'just chucking money' at a problem.

In this Institution there has had to be a major overhaul of the infrastructure. This is a long and slow process that does not make headlines but is about building fundamental foundations. If you do not have these, you will be dragging increasing amounts of 'old stuff' around with you, at an ever increasing cost.

IT infrastructure must endeavor not to become outdated or inappropriate. It is too expensive to try to work out what to do when change is needed. It also takes time — which businesses can usually ill-afford, especially in the financial sector. Overall, remembering to keep a focus on improving the infrastructure is the first lesson learned.

At the same time the IT world changing — and in three different directions. One concerns consolidation and the absolute emphasis on cost reduction or obtaining more for the same cost (or both). This is not unusual after a bout of large spending, but it is proving more painful than past 'corrections' because of its scale.

Then there is commoditization. This is spreading further and faster across both hardware and software than ever seen before. Think about blade servers and the power you can obtain in a small, inexpensive (relatively) box compared to (say) a mainframe or 'high-end' UNIX box. The cost ratio is proving to be highly beneficial for large users, especially when Linux is taken into account. The same is broadly true of storage. In software, it is the same again. While the Globus Toolkit Version 3 is not a product — yet, it will be. Furthermore, Grid computing is invading most software areas — from operating systems to high end middleware — that previously had been largely proprietary. There is real pressure on vendors to conform, or be left behind.

At the same time there are real indications that the way in which technology is going to be provided is about to change. The model that has been in use for so long — where any company of a medium or larger size has typically built its own IT infrastructure — is about to disappear. This does not refer to outsourcing, because that still retains an IT infrastructure that is recognizable as belonging to each organization — it just had someone else running it.

The lesson here is that about what the Grid has the potential to represent. The days of traditional IT are coming to a close. It is similar to what happened with electricity generation. Originally every factory possessed its own generator. Then utilities began to emerge and power provision was no longer a core competence. Specialized businesses — what we now call utilities — took over responsibility for distribution and provision.

The evidence indicates that this is just about reaching that point with computing power. Speciality utilities will have the advantage that they can provide computing power across a diverse collection of consumers — making money by exploiting the opportunities where not all customers are going to want computing at precisely the same time.

Arguably this started with Web hosting. Web Services offer the next spin of the wheel and the Grid could follow shortly after these. Part of the attraction of Web Services is that they are seeking to provide software services in a standard form. The consequence should be utilities providing computing power and enabling service-based solutions (including what have been middleware-type capabilities) that are sufficiently common that businesses of all sizes will want to use them as a utility.

What will happen is that commercial organizations will have access to standard middleware services that will enable access to common forms of transport (for example, request/reply or pub/sub or whatever). These will only be used between companies when it is sufficiently secure and when contracts can be negotiated — but the fundamental attraction will be great.

The inescapable lesson is that the bulk of most companies'

IT infrastructure and functions are going to become pretty much identical. IT infrastructure will increasingly become a specialized art, the province of fewer engineers — just as is the case in telco. or power or gas transmission.

For each individual organization, there are going to be some interesting discontinuities in IT. Deciding when to make the jump to utility computing/infrastructure is not going to be easy. But business considerations will slowly make it inevitable.

While it may have been unintended, the excesses of the late 1990s, when added to the turndown of the past three years and combined with Web Services and Grid-enabling technologies, are opening up a computing world where organizations will be paying others to provide IT infrastructures that previously would have been internally delivered. This will be a real wrenching change for existing IT. It will not happen overnight. But it is an aspect of strategic planning that is a real possibility.

Management conclusion

If you think that a more flexible, more connected infrastructure, leveraging and providing utility function, is the way ahead, positioning your IT infrastructure to support this model is vital to success.

The first thrust of this is providing layers of abstraction and virtualization which decouple implementations from each other. There are many layers of this (from IP to SAN, from VM to Container) but from a business point of view, this Institution believes that the two most crucial ones are the SOA 'bus' and the virtual data store. By concentrating on these two layers, business flexibility to acquire assets and leverage them and acquire customers and integrate them is enhanced.

Once you have layers of abstraction like this, the business can make decisions about which services are core competencies, which can be offered as utility services, and which should be bought from another utility.

Most IT professionals have spent a lot of time working on picking platforms, doing one-off integration projects and the like. In the new computing model, most of these will become esoteric issues mainly handled by the system itself. The challenge is to become one of the early adopters which manages to work out how to exploit the new, more cohesive, more intelligent infrastructure to gain business advantage, rather than being seen as a cost of doing business.

Implementing the UK's e-Science Grid

Dr. David Boyd
Deputy Director
CCLRC e-Science Centre
Rutherford Appleton Laboratory

Management introduction

David Boyd is the Deputy Director of the e-Science Centre at CCLRC which is based in the Rutherford Appleton Laboratory near Oxford and at the Daresbury Laboratory in Cheshire. The e-Science Centre has some forty staff, roughly two thirds in Rutherford and one third in Daresbury, whose main remit is to facilitate the development of a Grid middleware infrastructure in support of CCLRC's main science facilities and their users — like the ISIS neutron accelerator source at Rutherford and the SRS X-ray source at Daresbury.

The other major facility he supports through the Grid development initiative is the particle physics programme based at CERN in Geneva, specifically for the Large Hadron Collider that is due to go online in 4-5 years' time. This will generate very large amounts of data — several petabytes per year. This has become one of the major drivers for the Grid and related technological developments.

If these are his primary activities, Dr. Boyd also has a major role in developing and running the UK Grid. It is in this capacity that he shares his thoughts and experiences for the **MIDDLEWARESPECTRA** *(this first appeared in* **GRID.MIDDLEWARESPECTRA***, in the July 2003 2nd Edition — see page 51).*

The remit

Our remit here, in CCLRC, is to develop the application of the Grid within our own facilities. We also have been given the role of directing and supporting the practical development of the e-Science Grid infrastructure within the UK more generally.

This comes in two main ways. We are under contract to the Department of Trade and Industry and the e-Science Core Programme — to operate the UK Grid Support Centre. This provides the underpinning expertise — and support and other facilities — which users of the UK Grid need in order to be able to exploit Grid technology.

In this context, the UK's e-Science program has funded about 60 projects via the UK Research Councils and the UK e-Science Regional Centres (each of which has several projects). These projects are the customer base for the UK Grid Support Centre and we are there to provide:

- **technical support**
- **access to the necessary software**
- **provision of essential services**
- **the operation of the Grid**
- **the certificate authority which issues PKI/X.509 digital certificates to authorized participants in the UK e-Science Programme (thereby enabling them to use our Grid facilities securely).**

The last of these is essential. One major aspect of any Grid is operating a secure infrastructure. In the UK Grid case, we control access through the use of digital certificates. The Grid Support Centre is the trusted source for those certificates in the UK and we now have issued more than 400 certificates to participants in the UK e-Science program.

The second way in which we support the UK e-Science program is that we co-ordinate the activity called the Engineering Task Force. This is a consortium involving all of the UK's e-Science Centres (see Figure 2.1). One of the arrangements made, when setting up and funding these e-Science Centres, is that each one has had to commit to making available some computational resources for attachment to the UK Grid infrastructure. In other words each Centre has committed to install Grid middleware so that these resources can be connected together and operate as part of the UK Grid. In turn, this runs across Super Janet 4, which is the high speed private network which links the UK's universities, research and educational institutions.

The UK Grid Engineering Task Force, therefore, co-ordinates these activities, with the aim of bringing into reliable operation the working UK Grid. This is the basic Grid infrastructure which authorized scientists will be able to use to submit applications, to run jobs and to obtain results.

Progress and the Level One Grid

We have been working on the UK Grid in this way for well over a year. We started at the beginning of 2002 and the early stage work involved:

- **understanding the required technology base (in practice this is based on the Globus Toolkit)**
- **starting to investigate what are the issues involved in installation and operation.**

For example, we had to think about firewalls. Globus opens a number of IP ports so that remote machines can talk to each other through these ports. But, if machines are behind firewalls, this creates operational and management issues that have to be addressed by both network and security people if appropriate communications are to be provided.

To start doing this between a number of computing centres hosted in different universities you have to open discussion with the network people, the system managers and the security managers and explain to them:

- **what are the requirements of a working Grid**
- **why they are being asked to take actions (which, if there is a common first reaction, is usually 'we don't want to do this because of the associated risk')**
- **that there is an inherent security mechanism intrinsic in the Grid which will provide them with protection**
- **who will be able to use the opened ports**
- **etc.**

So our initial starting point was to understand both the technical and, to a degree, sociological challenges. We then had to identify, test and implement solutions. This was, therefore, very much a first enabling stage in the building of the UK Grid, which is why we called it the Level One Grid.

Around the summer of 2002 we reached the stage where we were able to run demonstration jobs across the UK Grid, between one Centre and another. Albeit somewhat cumbersomely, with manual intervention, we could enable jobs to be submitted and executed using the Globus MDS (the resource directory service) to locate the resources needed and then obtain the results.

The Level Two Grid

Having attained what we wished to do with the Level One Grid, we set ourselves a new target for April 2003. This was to implement and demonstrate a Level Two Grid.

This Level Two Grid aimed to move the degree of sophistication forward to the point where we could provide greater reliability and that various components were persistent to a much higher degree than before. The point here was that we would be able to carry out tests on a continuous basis as well as have a broader range of applications installed on the relevant machines which would:

- **allow clearer and more convincing demonstrations of Grid capabilities**
- **show how those capabilities could be applicable for delivering useful science.**

Let me explain what I mean about greater reliability.

In the Level One Grid we would have to call people at the other data centres and ask if we might perform a test at (say) three o'clock in the afternoon — and would the remote end ensure that the appropriate ports in any firewall were open and that resources were available for as long as was needed to do the test. This was, to all intents and purposes, a kind of 'manual automation'.

In the Level Two Grid we progressed. Our expectation was that we should do the required computing on a daily basis — using our Grid infrastructure. We expected (among other things):

- **the necessary components of the infrastructure to be available — for example, the Globus connection to local job managers**
- **for these to be running on each machine**
- **the file transfer protocol to be working**
- **required ports in firewalls to be maintained open.**

Once this became available we had a stable, operational, low-level infrastructure. With this we could start more sophisticated testing. This could then be on a daily basis — without needing special arrangements.

One of the big issues in any Grid is that its value comes from being able to operate end to end, from scientists in location A to scientists in location B (and with the appropriate computer resources). This means that all the different components in a Grid infrastructure have to be properly working and properly supported. It is much more than just accessing a Web site.

Indeed, we are still assembling all the software tools which will be required to run the Grid on top of the basic communications infrastructure. The Globus middleware, for instance, provides much of the necessary additional functionality that a Grid infrastructure requires.

Our Level Two Grid was successfully delivered in April 2003. We ran a number of demonstrations at events which showed a range of scientific applications running across our infrastructure.

The Level Three Grid

The Level Three Grid has the objective of moving what we know can work into a stable operational state where we transition from what is, essentially, a development environment to one that is in full operation. This requires us to move from what is still not particularly user friendly (although it is usable) to one that is much easier to access.

In addition, we intend to move from not actively promoting its use to a state where a wider community of users know it is ready for prime time. But to achieve this we need to gain a lot more experience of what operating a Grid involves — operating in the sense of the way we understand today how to manage a computing facility and how it is organized to run on a routine, production basis.

For this to happen we are currently researching:

- **what are the important operational parameters**
- **how to set them**
- **how to optimize performance**
- **how to diagnose and deal with problems as they occur**
- **how to establish appropriate procedures for a dynamic, distributed environment.**

We are trying to do this with software which is, if we are honest, still relatively immature. Most is based around the Globus Toolkit which, even though it is now much improved, is still middleware that is not really designed for production use.

There are also issues to do with authentication and authorization control. To define these:

- **authentication is being clear that we can vouch for the digital identity of a person**
- **authorization is saying, now that we know who a user is, what specific resources he or she can use.**

One part of this process is support for the mechanism which the Grid world calls the 'virtual organization'. An important aspect of the Grid is that it can support the operation of ephemeral collections of people who come together to address some particular challenge. These people do not necessarily always belong to the same organization or exist in the same physical location. But they do want to co-operate, with a minimum of obstacles, and share resources and work together. These collections of similarly tasked people are the virtual organizations.

So, in deciding who can obtain access to which resources, the key test is — of which virtual organization is a user a member (remembering that any individual can be a member of one or more virtual organizations). In this context we need a way to assign roles to people so that they can access the resources not on the basis that they are a particular person, but on the basis that they are 'a member of a particular environmental science Grid project team' or whatever. We need ways of validating this, and we also need ways of implementing and enforcing access policies from the perspective of the resource owner.

The decision about whether people (or applications) can use specific resources must finally rest with the resource owner (or manager). There has, therefore, to be a mechanism whereby each resource owner can implement access policies and control access so that the resource owner can say, with confidence, 'OK, I recognize that this person is in this virtual organization and that I am I going to allow that virtual organization to use this part of my resource'.

Authorization is a complex process within the environment of a Grid — where you have to support the different required tasks with good tools. This is not the scenario we have reached yet. On the other hand, we have experimental tools and we are using these to gain experience of the practical issues.

In the Level Three Grid we will be pushing the capabilities and testing ever more thoroughly so that we can run in more stable and proven ways. That said, I am in no doubt that the operational experience of running our current Grid will give us new insights into the practical parameters which really matter. Some of these will be quite unforeseen, while others are more obvious:

- **how do you de-bug jobs when they fail?**
- **what are the different ways in which things can not work properly?**
- **do we have diagnostic tools to understand the sorts of problems that arise?**
- **do we possess the performance tuning tools that allow us to optimize overall Grid (as well as individual component) performance?**

Figure 2.1: the UK e-Science Grid

- **depending on the different kind of job mixes, and other operational parameters, are we providing enough information to users (of our Grid) to enable them to exploit it effectively?**

Applying Grids

In our UK Grid arena the people who have the largest imminent problem are the particle physicists. The reason is simple. With the Large Hadron Collider (LHC) due to come online around 2007, in Geneva, they have to solve the problem of how to process the huge amounts of data it will create. They are moving forward with the Grid because it is an imperative for them.

This is probably not true, to the same degree, of any other science area at the moment. Rather, most people are using the UK Grid because they see it as a possible way of improving their capabilities for addressing important science problems, rather than being unable to live without it.

The particle physicists are experimenting with various Grid testbeds. They are building the LHC Computing Grid which involves resources at centres across the world — because the LHC is a global facility. The applications that they are currently running include Monte Carlo simulations of how events will behave in the detectors in order to develop the data analysis programs they will need when LHC starts up.

Elsewhere, we have the astronomers working with the UK Grid on a project called AstroGrid. This is developing the concept of the virtual observatory. This has some interesting implications, not all of which were expected.

The current situation is that there are many astronomical facilities around the world — optical telescopes, radio telescopes and satellites — which are collecting data, often at specific wavelengths. These instruments are filling databases with 'information' which astronomers can access.

However, if you want to pose a question like 'can you tell me all the information known about this astronomical object?', you may have to search multiple databases, each of which will likely have a:

- **different access protocol**
- **different internal data format**
- **different ways of presenting results.**

To the astronomer, overcoming this is time consuming. It limits progress. In fact it is not that different, I am told, from what occurs in the commercial world — where information needed by organizations can be held in many equally different representations...

What the virtual observatory is trying to do is to provide a layer of integration that sits above all of those existing data resources — plus new ones that are being developed, like the Vista telescope — to provide a homogeneous interface to an astronomer. Once in place, this will enable astronomers to pose questions in ways that make more sense from an astronomical perspective.

In effect, astronomers will be able to submit a query to a range of resources in different places. All of the heterogeneity will be made invisible via the meta data layer. The results obtained will have integrated the data that is available in all of those facilities that are relevant to the question asked. What is so interesting about this is that the astronomer may obtain all sorts of data that he or she was not expecting — if only because he or she was not even aware that somebody else was collecting data on a different frequency or nearby object or whatever.

The idea of calling it a virtual observatory is that traditionally, when an astronomer wanted to do an experiment, he or she would have to arrange time at a real observatory, collect the data and then analyse it. In the future we anticipate that astronomers will use the virtual observatory to ask their questions. Only if data is not available will telescope time have to be booked.

Furthermore, astronomy telescopes will be used better because — before booking telescope time — the scientists will know what data already exists. Only that which does not, need be researched.

The UK Grid is also involved in environmental science projects. These involve people who are doing large simulations of the ocean and of the atmosphere over extended time periods, perhaps a hundred years, and looking to understand how the earth's climate will evolve. If you are looking at climate change over a period of hundreds of years into the future you need access to massive computing resources as well as access to as much data as is already available from previous simulations plus measurements taken from the atmosphere and the oceans over an extended period in the past.

It is the integration of all this data that allows environmental scientists to compare observational data with model data and thereby refine and improve the quality of their models and the quality of the predictions coming out of those models. That should mean better climate predictions for all of us.

The fourth application area I must mention is one that is likely to have even more significance for the future. This is in the biological science area. Through a range of experimental processes happening now we are gathering data on genomes for a range of organisms. The human genome attracts much attention but there are people studying insects, plants and other vertebrates. Researchers are building databases on genes and proteins, including protein structures and protein functions.

Partly through experimental facilities — such as the Synchrotron Radiation Source currently operating at Daresbury and the Diamond Light Source being built at Rutherford — we are going to be accumulating vast amounts of new data which will enable researchers to understand the structure and function of these very complex molecules involving tens of thousands of atoms. These are, essentially, the building blocks of life.

These protein databases are a primary research resource. Somebody looking to develop a drug for a disease or disorder may well want to search through known protein structures to find out how these would respond to interactions with (say) new or existing drugs. These interactions can then be modelled in silico rather than tested in a laboratory or clinical setting — which is hugely cheaper as well as much faster.

In such ways, the Grid is a mechanism for bringing both computational and data resources together to enable people to investigate complex problems. From a human point of view, this work may yet have a greater long-term impact than areas like particle physics and astronomy which were the initial stimulus for developing Grid technology. I think the potential of the Grid to improve the biological sciences, and our understanding of how organisms, cells and the whole structure of life works, has enormous potential for the future.

Sociological and management implications of Grids

In talking about the sociological and management implications of Grids, I should start out by stating that what I am about to discuss is my personal view. That said, I think that Grids do, quite consciously and deliberately, cut through traditional organizational and resource management boundaries. Grids are all about linking together resources which are:

- **owned by different people**
- **operated in different places**
- **inherently heterogeneous, both technically and organizationally, for some common purpose.**

That requires a change of mind set. The traditional view is that local resources are (in general) operated in optimal ways to satisfy the local requirements of local users. Then along comes the Grid which espouses the virtues of accepting work from people outside the local environment, whom you may never have met (and may never meet). These people, whom you do not know, are going to use your resources. Furthermore the Grid is going to:

- **verify that they are who they say they are**
- **assert that their purpose is honorable**
- **monitor their behaviour (so that if misbehavior occurs, it can be caught and dealt with appropriately).**

This is a very different scenario for system administrators and data resource managers. Scientists may appreciate the research opportunities created. But that does not remove the sense of challenge from the administrators and managers, the people who are responsible for the day-to-day operation of these computing and data resources.

These are issues to do with change. They are not unique to e-science or even commerce. The important dimension is that better science, in our case, is the intended result.

But communicating this to people who may see these changes as a threat to the integrity of the resources they manage is not always easy. Remember that we are dealing with different roles. These are the people whose job it is to run computing facilities in a secure and reliable way. They are not scientists: they are system managers. It may not be top of their agenda to make 'their' resources more accessible to a much wider range of users in ways which, to them, seem significantly to weaken their current security measures. Remember: they are the people who are responsible for the integrity of these resources.

At the same time, do not assume that all scientists are automatically in favour (of Grids). I have met those who say, 'this is my data which I have worked very hard to collect; only I am going to analyze this and I don't want anybody else to have access to it, now or even possibly for ever.'

One of the less obvious consequences of a Grid is that it exposes issues to do with who actually owns data. For example, if you are employed by a university or a research institution, does the data from your research belong to:

- **you?**
- **your employer?**
- **a Research Council (in the UK) which gave you the money to do the experimentation?**
- **the taxpayer, who paid the taxes that funded the Research Council?**

Another way to look at this is to ask whether the data collected in all publicly funded research should be in the public domain or if access to it may be arbitrarily restricted. Science has traditionally been an open activity where knowledge is internationally shared, but does this extend to sharing primary research data? Should another scientist who has done some experiments and wants to put his data together with yours (to see if he can learn something new) be freely able to do this, if necessary without your permission?

These are the kind of issues that the Grid is exposing. It is happening because Grids enable possibilities that were so hard before that people just did not try to do them.

Now we are saying that, if all data resources are stored online as they increasingly are, the Grid can provide much more access transparency. Potentially you will have people asking the question, 'well, why can't I see all the other data that environmental scientists have collected in the UK?'. That today may not be too sensitive a request. But if you go on and say 'well, I want to look at all the biological protein diffraction results from people who have been looking at structures of proteins in the UK', you will get a very apprehensive response — not least because there are potentially huge commercial rewards at stake.

There is a major issue here which, in my personal view, has not been sufficiently well addressed to make it clear how the Grid can be used. Grids raise questions about ownership and accessibility. For the future, we could simplify the situation — by making it a condition of all (say) publicly funded research that the primary research data are placed in the public domain. But that does not solve everything. There remains the legacy problem, not least that it costs a lot to make this data available online.

Simultaneously, we are looking at experimental science becoming increasingly data-rich, because new equipment is becoming higher resolution in space and time, with more data being collected with greater precision. This applies to almost all experimental facilities. We must face the fact that we going to have more and more data. We need to start to understand how to manage that data for the best use of the whole science community.

These are some of sociological and management issues that, in my personal opinion, the UK research community (or most anyone else, for that matter) has yet to get to grips with. We have bodies funding the majority of public science in the UK and investing much public money into the creation of research data. But we lack a framework for how that data should be stored, looked after and made available. Grids will, eventually, force us to address these issues — but it would better if this were done sooner rather than later.

Lessons learned

My first lesson learned from my experience with the UK Grid is — engage all interested parties in understanding the issues that are associated with Grids. There is a lot of education to be done before you can start making technical demands of people. I am thinking particularly of the interface between computing system managers and Grid developers, which is a specific area where I have been particularly working to try and break down barriers. If the desired solution involves a number of parties with different responsibilities, technical expertise and interests, gathering them together and understanding each others' needs, and agreeing to co-operate in addressing those needs, is vital.

Grids, because of their nature, can seem to accentuate divisions — unless these have been addressed beforehand. If we are to succeed in providing a working Grid infrastructure, we must transcend different interests in different communities.

My second lesson is that we need to appreciate that we are trying to build something stable and useful from a user's perspective when what we have to work with is still pretty immature and a relatively buggy set of software that does not work particularly well together. Grid software is not terribly well integrated and needs much more work done on it.

Having said that, it is also clear that the Grid is accelerating the normal technology development cycle. What would have taken years in the past we are now doing three or four times faster. We do not have the luxury of having all the bits work before we try and build the next stage.

Understanding the implications of this accelerated technical development — and the associated management of operations, risk and the problems that arise in such an environment — is both stimulating and challenging. In some ways I cannot believe how far we have come in two years, given how little we had when we started. To me, the UK Grid is an achievement already.

Management conclusion

In this discussion Dr. Boyd offers a view of the UK Grid, its early stages of evolution and several of the issues that he sees emerging — not least about the ownership of data and intellectual property. He shows how much has been achieved in the UK science community in a short space of time.

What is even more remarkable is how many of these issues are directly applicable and relevant to the commercial world. Those wishing to introduce Grids in the commerce world would do well to learn from what he and his colleagues, under the UK e-Science umbrella, are achieving.

Web Services orchestration and choreography

Tom Welsh
Consultant

Management introduction

Five years have ticked by since the idea of Web Services was first mooted. Three years have passed since Microsoft launched its ambitious .NET architecture, which depends heavily on Web Services. Those years have witnessed an energetic drive by IBM and Microsoft, working more closely than at any time since their OS/2 debacle, to:

- ***popularize the new Web Services paradigm***
- ***establish a universally accepted standards 'stack'.***

With every additional specification published, the original vision of simplicity and ease of use has been further diminished, even tarnished. XML, XML Schema, SOAP, WSDL, UDDI, WS-Inspection, WS-Security, WS-Coordination, WS-Transaction, WS-Reliable Messaging, WS-Addressing, WS-Policy, WS-Referral, WS-Routing... the list goes on and on. To insiders, it may seem straightforward; for the rest of us, we should be forgiven if we feel mounting bafflement.

The latest proposals deal with a yet another challenge: how to define business processes and transactions in terms of Web Services. When a purchase order arrives, for instance, what steps should be taken next? A reply must be sent, of course, but perhaps other business processes need to be started; certain parameters should be

checked (such as the size of the order); and business rules or constraints will likely apply.

In an ideal world, business professionals and managers will be able to specify these reactions and responses themselves, without having to bring programmers into the picture. This ambition has come to be known as 'orchestration' or 'choreography': the analogy with the way in which skilled musicians and dancers co-ordinate their actions while carrying out predetermined instructions is entirely deliberate. The possibilities are entrancing... but at this very moment new difficulties appear.

In this analysis, Tom Welsh looks at both choreography and orchestration, including such alphabet soup initiatives as BPEL4WS, XLANG, WSFL, WSCI, WSCL and BPMI.

Orchestration, choreography, work flow... what is it all about?

The critical element that sets orchestration and choreography aside from previous specifications — like the Web Services Definition Language (WSDL) — is timing. WSDL can give complete details of how a Web Service should be invoked, and what information it requires and returns. It cannot, however, convey a constraint such as: 'the registration Web Service must be successfully invoked before any purchase Web Services can be used'. In other words, some kind of work flow is necessary if Web Services are to be strung together and sequenced in a logical order.

"The full potential of Web Services as an integration platform will be achieved only when applications and business processes are able to integrate their complex interactions by using a standard process integration model." That is how the Business Process Execution Language for Web Services (BPEL4WS) 1.1 specification sums up the case for choreography.

Until recently, most experts would have said that orchestration and choreography were just different ways of conveying the same idea. Now — perhaps because they have been thinking much harder about exactly what these terms mean — some vendors have started drawing a distinction between them. Most experts who make this distinction agree that:

- **choreography is higher-level, more abstract and top-down — whereas orchestration is lower-level, more concrete and bottom-up**
- **choreography is declarative (in other words, it states the desired outcome, rather than saying exactly how that outcome is to be accomplished) — whereas orchestration provides the specific details of how business processes are implemented.**

It follows that, among other things, a single piece of choreography could be fulfilled by several different orchestration solutions.

Standards consortia

Although several consortia are involved with Web Services, there are really only two important players at this stage (other bodies have important roles to play, but stay in the background):

- **W3C**
- **OASIS.**

W3C was first in the field, accepting the submissions of Simple Object Access Protocol (SOAP) and WSDL back in 2000. OASIS staked its claim by accepting responsibility for UDDI and WS-Security in 2002, but since then it has hardly been out of the news. Some commentators believe that IBM and Microsoft, the two dominant Web Services vendors:

- **have lost patience with W3C**
- **think that OASIS can provide them with a faster, better 'service'.**

"OASIS and the W3C are very different places altogether," says Tim Berners-Lee, the chairman and co-founder of W3C. "The rules are very different. [At the W3C] a lot of that has to do with getting everybody on board, making sure everything is co-ordinated and trying to get the standard very widely deployed. [We require] a demonstration of implementation, of interoperability, before something can become a standard. We have public review. We have requirements that the groups be chartered to liaise with groups which have related technology.

"What OASIS provides is somewhere you can start an activity with no prior requirements. Three companies can just get together, they can start a group, and there's nothing to prevent the same thing [from] being done in different organizations. There's nearly no management control at all. It's faster."

It looks as if control of Web Services standards may be split between W3C and OASIS. That is not necessarily bad. The whole apparatus, comprising over two dozen specifications, is probably too much for any one body to handle. Besides, the split represents a logical division of labor:

- **W3C looks after the 'plumbing'**
- **OASIS concentrates on the higher-level (and, arguably, more contentious) 'business process' standards.**

W3C

The World Wide Web Consortium (W3C) was founded in October 1994, on the initiative of Tim Berners-Lee (the inventor of the Web) who remains its director today. One of the motives for setting up W3C was to preserve the Web from fragmentation under commercial pressures.

In January 2002 the W3C's XML Protocol Activity was subsumed into a new, more extensive Web Services Activity. This comprises a co-ordination group and three working groups:

- **Web Services Architecture**
- **XML Protocol**
- **Web Services Description.**

The Web Services activity embraces more than 40 diverse Activities in which W3C is presently engaged. Others include HTML, Hypertext Transport Protocol (HTTP), Semantic Web and XHTML. No fewer than 11 Activities are devoted to aspects of XML.

W3C was tracking various companies' efforts to develop orchestration standards as early as 2001, but the issue did not move into the limelight until the Web Services Choreography Interface (WSCI) was submitted in June 2002. Three months later Oracle began proposing that W3C create a dedicated working group to pull together industry thinking about orchestration and choreography.

The Web Services Choreography Working Group was established in January 2003, and Oracle hosted its inaugural meeting. The Working Group's charter commits it to an initial two-year period of study. This is the minimum period before a formal 'Recommendation' can emerge, although experience suggests that results can take longer.

To start with, the Choreography Working Group will look at WSCI and HP's Web Services Conversation Language (WSCL). In this context, BPEL4WS is currently ruled out on at least two grounds:

- **it has not been submitted to W3C**
- **there is still thought to be some uncertainty about its royalty-free status.**

The Working Group's first meeting touched off a flurry of media attention, arising from the story that Microsoft sent two delegates who made an important contribution, but never returned. Steve Ross-Talbot, chief scientist at Enigmatec and co-chairman of the Working Group, said he was surprised and disappointed by this outcome.

"As co-chair of this working group I am totally mystified as to why Microsoft has decided to withdraw from the group. When Dr. Meredith and Dr. Brown attended, during the face-to-face last week, they both made outstanding contributions to the group in a very short space of time" Ross-Talbot said in an e-mail. "They presented a position relative to BPEL4WS, of which Microsoft is a co-author along with IBM and BEA, that was totally in keeping with the stated focus of this group as per the charter."

Whatever the reason, IBM and Microsoft have not graced the proceedings of the Working Group with their presence since that first day. But at least nine companies — BEA, EDS, Intalio, Novell, Oracle, Sun, TIBCO, SAP and SeeBeyond — are members of both the W3C Choreography Working Group and the OASIS WSBPEL Technical Committee. That fact holds out at least some hope of future co-operation, especially as heavy hitters like Oracle and Sun are publicly committed to seeking reconciliation between the two committees.

OASIS

The Organization for the Advancement of Structured Information Standards (OASIS) was founded in 1993 under the name SGML Open. Its original mission was to foster interoperability between products supporting Standard Generalized Markup Language (SGML). When W3C created XML, essentially an enhanced subset of SGML designed for use with the Web, SGML OPEN changed its name to OASIS in recognition of the need to address a wider range of languages, including XML.

OASIS describes itself as driving "the development, convergence and adoption of e-business standards". It has drawn up standards for:

- **security**
- **Web Services**
- **XML conformance**
- **business transactions**
- **electronic publishing**
- **topic maps**
- **interoperability within and between marketplaces.**

In addition, OASIS has a close working relationship with the United Nations Centre for the Facilitation of Procedures

and Practices for Administration, Commerce and Transport (UN/CEFACT). This has a mandate which covers worldwide policy and technical development in the area of trade facilitation and electronic business.

In addition to the ongoing Electronic Business XML (ebXML) initiative, OASIS has recently begun taking a lively interest in Web Services. It now has half a dozen technical committees (TCs) devoted to topics such as Web Services management, security, reliable messaging and choreography.

The last-named got under way in April, when OASIS set up a TC to discuss the standardization of Web Services Business Process Execution Language (WSBPEL). The BPEL4WS 1.1 specification was submitted to this TC by BEA, IBM, Microsoft, SAP and Siebel. To some people's surprise, Intalio, Oracle and Sun joined the TC despite being advocates of WSCI and members of the W3C's Choreography Working Group.

The specifications

Because the whole 'Web Services work flow' space is unexplored territory, its borders are (at best) blurred. XLANG, WSFL, BPEL4WS, WSCL and WSCI are definitely concerned with orchestration, choreography or both.

But what about BPML, ebXML and the various already existing work flow standards? Only time will tell. In the next sections, I will look briefly at the most important specifications of today.

XLANG

Microsoft announced XLANG for BizTalk Server in July 2000, along with the SOAP Contract Language (SCL) and SOAP Discovery (Disco). XLANG was already fully supported by BizTalk, being:

- **automatically generated by BizTalk's graphical Application Designer**
- **executed by the BizTalk Orchestration Engine.**

According to the original specification published by Microsoft, "...the goal of XLANG is to make it possible formally to specify business processes as stateful long-running interactions. Business processes always involve more than one participant. The full description of a process must show not only the behavior of each participant, but the way these behaviors match to produce the overall process. The focus is on the publicly visible behavior in the form of messages exchanged. Each participant clearly implements its behavior using some private means. The details of these private implementations are not a part of the business protocol, and XLANG provides no means to specify them. XLANG aims to specify all the behavior and only the behavior that the participant explicitly wants partners to understand in designing their own service processes. The specific high level feature categories that define the scope of XLANG are listed below:

- **sequential and parallel control flow constructs**
- **long running transactions with compensation**
- **custom correlation of messages**
- **flexible handling of internal and external exceptions**
- **Modular Behavior Description**
- **dynamic service referral**
- **multi-role contracts."**

XLANG has now been superseded by BPEL4WS. This is the choreography language agreed on by IBM and Microsoft that was recently submitted to OASIS for standardization.

WSFL

Unveiled by IBM in May 2001, Web Services Flow Language (WSFL) "is an XML language for the description of Web Services compositions. WSFL considers two types of Web Services compositions: The first type (flow models) specifies the appropriate usage pattern of a collection of Web Services, in such a way that the resulting composition describes how to achieve a particular business goal; typically, the result is a description of a business process. The second type (global models) specifies the interaction pattern of a collection of Web Services; in this case, the result is a description of the overall partner interactions."

Its background comes from years of IBM experience with products like Flowmark and MQSeries Workflow — both of which are traditional business work flow suites aimed at large organizations. As such, WSFL has not been superseded by BPEL4WS, in the same way that XLANG has; IBM still has products in use where WSFL is a key component. But that is not to say that BPEL4WS could not take over from WSFL in due course.

BPEL4WS

BEA, IBM and Microsoft announced Business Process Execution Language for Web Services (BPEL4WS) in August 2002, together with WS-Coordination and WS-Transaction. BPEL4WS was positioned as the convergence of Microsoft's XLANG and IBM's WSFL, which seemed suffi-

cient to warrant it being regarded as the likely unchallenged market leader.

The role of BPEL4WS is to define new Web Services by composing a set of existing services. In so doing it can:

- **perform computations**
- **check conditions**
- **handle exceptions.**

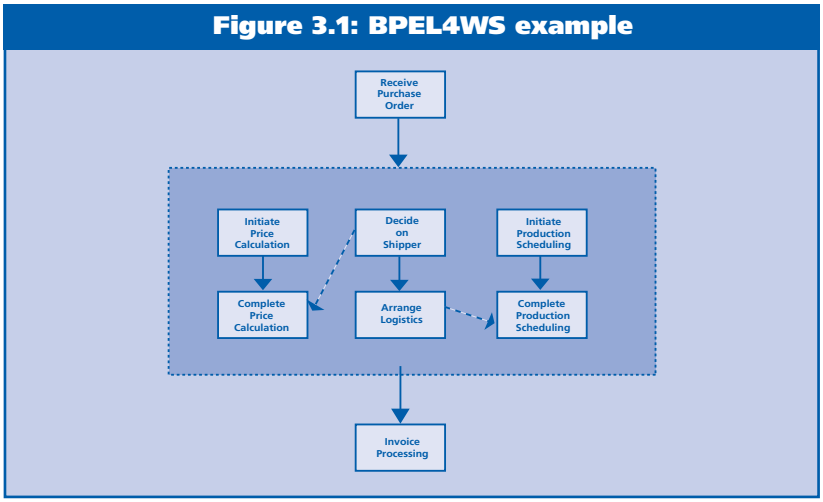
The interface of a composite service is described by a collection of WSDL port types, just like an ordinary Web Service. To illustrate, Figure 3.1 shows a typical business process scenario that might be handled by BPEL4WS (the straight lines represent sequencing; free grouping of sequences represents concurrent sequences and the dotted arrows represent control links used for synchronization across concurrent activities).

On receiving a purchase order from a customer, the process (in Figure 3.1) initiates three tasks in parallel:

- **calculating the final price for the order**
- **selecting a shipper**
- **scheduling the production and shipment for the order.**

While some of the processing can proceed in parallel, there are control and data dependencies between the three tasks. In particular, the shipping price is required to finalize the price calculation, and the shipping date is required for the complete fulfilment schedule. Only when the three tasks are completed can invoice processing proceed and the invoice be sent to the customer.

Figure 3.1: BPEL4WS example



BPEL4WS and WSCI

IBM maintains that WSCI is a proper subset of BPEL4WS. Because of this, it argues that there is nothing to gain from seeking to merge the two. Furthermore, IBM says that BPEL4WS provides full support for long-running transactions — through its integration with WS-Coordination and WS-Transaction. In contrast, WSCI lacks transaction processing support.

Critics soon noted that XLANG and WSFL were distillations of Microsoft's BizTalk Server and IBM's WebSphere MQ Workflow respectively. They have argued, with some merit, that this blend of the two would inevitably be overly vendor-specific.

Nevertheless, such objections were always likely to be swamped by the sheer marketing weight of the combination of BEA, IBM, Microsoft, SAP, Siebel and their 22 co-submitters. Other leading Enterprise Application Integration (EAI) vendors — for example Vitria and webMethods — were quick to follow suit. Several commentators have expressed the hope that BPEL4WS will turn out to be as easily implemented by vendors like Avinon, Collaxa, Intalio and Iona as by BEA, IBM and Microsoft.

Shortly after announcing the BPEL4WS specification, IBM released a Java implementation called (surprisingly enough) BPWS4J through its alphaWorks Web site. It also provided a plug-in for the open source Eclipse IDE, on which its own WebSphere Studio Application Developer (WSAD) is based.

In the longer term, IBM plans to add support for BPEL4WS to its WebSphere Business Integration product. This is currently a heterogeneous collection of tools including WebSphere MQ Workflow, WebSphere MQ Integrator and InterChange (the process automation package it acquired with CrossWorlds in 2001). IBM has also announced that it will rewrite WebSphere Business Integration to become J2EE-compliant.

Clearly this is a major commitment which had to be planned well in advance. IBM would be reluctant to contemplate any changes in the status of BPEL4WS at this stage, as that could mean going back to the drawing board and a lengthy delay in getting to market. No doubt Microsoft feels the same about its plans to incorporate BPEL4WS in BizTalk Server 2004.

BPML, WSCI and WSCL

The Business Process Management Initiative (BPMI) was initiated by Intalio and set up by 16 companies — including BEA and Sun — in August 2000. HP and IBM joined later, accompanied by system integrators like Accenture, Cap Gemini and KPMG. The first public draft from the BPMI was the Business Process Modeling Language (BPML) which was released in July 2002.

BPML is an XML-based meta language for modeling business processes. It uses WSCI to define public interfaces to those processes. BPML has two practical points in its favor:

- **it provides a user-friendly graphical design interface**
- **it is directly executable — obviating the need for a slow (and error-prone) development phase.**

According to the BPMI, BPML is "a proper superset" of BPEL4WS. In an aggressive position paper dated August 2002, it declared that: "Microsoft pioneered the adoption of the Pi-Calculus model with XLANG, IBM rejuvenated the use of Petri Nets with WSFL, and BPMI.org unified the two approaches with BPML 1.0".

BPML was quickly taken up by vendors like Fujitsu with I-Flow, and Popkin Software with its System Architect modeling tool. But the anticipated widespread adoption has not happened, no doubt because hard-headed product managers continue to expect BPEL4WS to sweep the board.

BEA, Intalio, SAP and Sun beat IBM and Microsoft to the punch when they announced Web Services Choreography Interface (WSCI) in June 2002. While XLANG and WSFL had already been in active use for two years by then, the big two did not get around to publishing BPEL4WS until two months later. The WSCI authors stated that "after a public review period, WSCI will be submitted, on a royalty-free basis, to an industry standards body". The review period was minimal, as WSCI 1.0 was submitted to W3C before the end of June 2002. BPMI.org, Commerce One, Fujitsu, Iona, Oracle and SeeBeyond joined the WSCI authors in this submission.

The launch of WSCI may have been the first overt sign of rebellion against the dominant IBM-Microsoft partnership. Everything up until then — SOAP, WSDL, UDDI, WS-Inspection, WS-Security — had been initiated and controlled by the two industry leaders.

HP submitted Web Services Conversation Language (WSCL) to W3C in February 2002. Along with WSCI, it is being studied by W3C's Choreography Working Group. WSCL "fills the gap between mere interface definition languages that do not specify any choreography and more complex process or flow languages that describe complex global multi-party conversations and processes". Moreover, it "allows the abstract interfaces of Web Services, for example, the business level conversations or public processes supported by a Web Service, to be defined".

Management conclusion

The Web Services standards 'stack' has already become disconcertingly complex, although small subsets of it (such as SOAP and WSDL alone) can be used successfully for many applications. The full panoply of specifications is required only for inter-enterprise e-business transactions.

As for orchestration and choreography, these really only come in to focus when two or more organizations wish to embed long-running interactions within their business processes. This would be a difficult and complicated business no matter how implemented. Web Services standards such as BPEL4WS aim to streamline it as far as possible.

At first sight, the 'schism' between W3C and OASIS in the matter of choreography standards is distressing (or business as usual in the IT world). In practice, matters may not be so bad as they initially appear. There is a natural division of responsibilities between the two consortia:

- ***W3C owns the Web services infrastructure***
- ***OASIS is dedicated to e-business.***

By that logic, choreography should belong to OASIS. This argument is reinforced by the consideration that BPEL4WS, with the combined backing of IBM and Microsoft, is likely to prevail in the business (process) marketplace.

Finally, BPEL4WS is a complete solution based on two specifications (XLANG and WSFL) that have both been reasonably tested when in production. None of its rivals can plausibly claim any decisive technical superiority. So, although only a minority of organizations will have the resources to tackle choreography in the immediate future, those that do should not be faced with any agonizing decisions.

Java breaks new middleware ground

Dr. Keith Jones
IBM Software Solutions Worldwide

Management introduction

There is little doubt that Java has made a significant impact on IT infrastructures around the world in the eight years since its introduction. At JavaOne 2003 the Java community met to celebrate extraordinary deployment figures for Java systems, and to review developments for the coming year. New ground has indeed been broken in the Java architecture for business systems.

In this analysis Keith Jones reviews the work being done by the growing Java community to define and absorb open standards into the architecture behind Java systems. As the architecture is developed so too is the business value that results from deployment. In addition, he addresses the questions:

- ***is Java middleware about to reach its limits?***
- ***will complexity stifle its innovation before long?***

Java everywhere

At JavaOne this year a surprisingly large crowd of enthusiasts made the annual pilgrimage to San Francisco to learn about the latest Java developments and exchange the latest Java insights. This was surprising, not only because the economic climate has dictated stringent travel expense constraints but also because, after eight years, you might think that enthusiasm for Java would have waned.

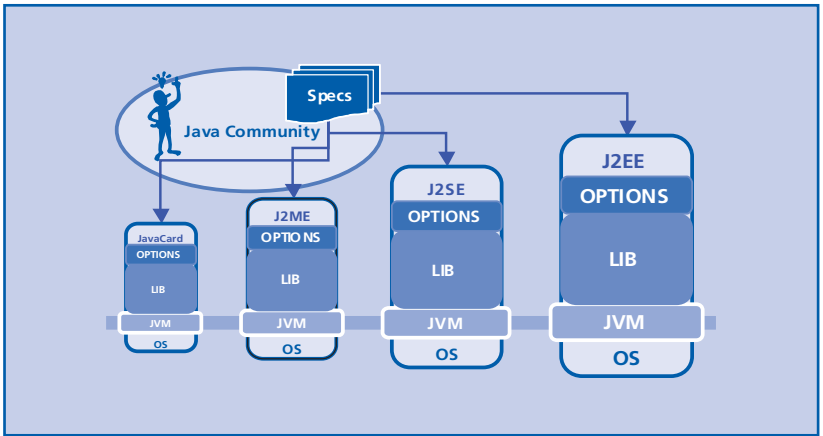


Figure 4.1: Java platforms

Not so — the Java Community continues to grow at an impressive rate as does the value that Java delivers. There are more than 3 million Java developers around the world creating that value — many more than either VisualBasic or C/C++ according to recent surveys. The projection is that this number will increase in the foreseeable future to more than 10 million.

One reason for such a projection is that Java has evolved from a uni-dimensional discussion about a programming language to a discussion about an entire system — a system with a single architecture that satisfies the needs of widely diverse users. Figure 4.1 shows the four platforms that share that single architecture. Each has:

- **a Virtual Machine (JVM) supporting the Java language**
- **a standard set of libraries**
- **optional functional extensions.**

The importance of the Java Community

The success of this architecture reflects the contributions from the many vendors, developers, enterprise users and consumers around the world who constitute the Java Community. For example:

- **over 300 million secure Java smart card devices have already been deployed**
- **more than 150 models of wireless device (mobile phones, PDAs, communicators) are Java- enabled (J2ME) and are now being built at the rate of 40 million units per month.**

It has been estimated that more than 500 million personal computers are running Java (J2SE) applications. Many of these are connected to computers at Fortune 500 companies which use Java (J2EE) platforms. 97% of all application servers deployed around the world run Java applications. By any reckoning, the sheer volume of processors running Java today is impressive — and the numbers continue to grow.

The power behind this phenomenon is the Java Community — inspired by the inventors of the Java language. Working in expert groups, this Community has been able to define an expanding set of standard APIs and functions that encompass the current specification levels for each of the four Java platforms. The many software vendors that collaborate to define specifications within the Java Community also compete as they bring compliant products to market.

Consistency across the Java platforms has always been important. Interoperability between those platforms has now become a focus area for the community as increasing numbers of diverse J2ME client devices are connected to J2EE servers.

The Java Community has provided a rallying point for vendors and users alike. Its members have defined an expanding marketplace for Java applications and middleware that probably could not have been developed by any single vendor in the same time-scale.

Evolving Java middleware

The majority of Java middleware products have been developed to support the J2SE and J2EE specifications. By far the most valuable architectural components built into those middleware products have been:

- **the Java Virtual Machine**
- **containers for Java components**

- **APIs that map relatively easily to Web servers (for presentation) and existing data sources (messaging systems and other enterprise information systems).**

In Figure 4.2 the de facto development phases for open standard Java middleware are illustrated. Leading edge vendors conduct research projects that result in technology prototypes. These are often made generally available for evaluation by academic, open source and enterprise organizations. For example, the Apache AXIS and WSIF projects started this way. Many other examples can be seen on readily accessible Web sites (for example, IBM's Alphaworks Web site).

Technology prototypes are often the inspiration and sometimes the substance behind standards proposals. Such proposals undergo scrutiny and development by a community of experts drawn from a relatively wide variety of backgrounds and interests. IBM's WSFL and Microsoft's XLANG are two such technologies that have recently been merged into a single proposal — BPEL4WS — that was then submitted for further development and adoption to an OASIS technical committee.

As standards are developed, and later finalized, middleware vendors ready product implementations for general availability. This phase often takes 18 months to complete. SOAP 1.2 is a recent example. A W3C standard is now emerging from this phase, with middleware vendors preparing their products for market. SOAP 1.2 is also an example of a standard that will be implemented in combination with others (like WSDL 1.2) to provide a working platform for future application systems.

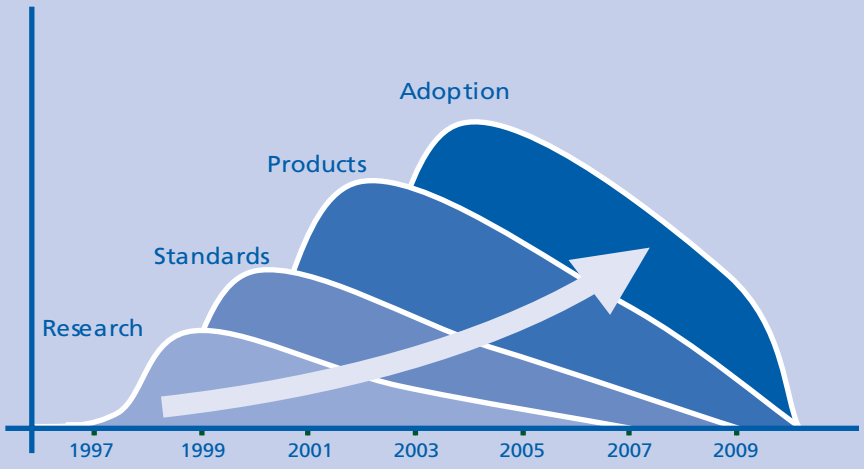
J2EE compliant platforms must include technologies built to conform to multiple open standards in combination. The J2EE specification provides a checkpoint (Figure 4.3, on the left) in the open standard middleware development process by requiring incorporation of agreed levels of underlying standards. Using standard compliance test suites, vendors are able to illustrate their products' felicity. This has worked well up to and including the current J2EE version 1.3.

The open standard approach taken by the Java Community has, therefore, resulted in a set of middleware products that deliver modern infrastructures for Web and, increasingly, transactional applications. Standards have facilitated both choice and competition —- between Java vendors and their platforms.

WS-I

But interoperability between Java and non-Java platforms for certain applications — those that use Internet communications — has required a new, higher, level of check-

Figure 4.2: Evolving Java middleware development phases



point. There are now 160 companies that have combined to form the Web Services Interoperability organization (WS-I.org) specifically for this purpose.

The proposed WS-I Basic Profile (Figure 4.3, on the right) is expected to be finalized in the third quarter of 2003. It not only requires compliance with specified levels of the standards listed but also identifies more than 200 usage constraints that must be observed in order to achieve interoperability between Web Services platforms. In light of this, the Java Community has temporarily delayed the final version of the J2EE 1.4 specification in order to support WS-I interoperability at the Basic Profile 1.0 level.

Business value

As architects we often address a problem domain with a layered approach, such as the one shown in Figure 4.4. The value of such an approach is that layers of abstraction permit us to expose relationships and dependencies in advance so that the style of solution eventually chosen will:

- **satisfy functional requirements**
- **achieve resource efficiency**
- **be positioned, for maximum potential value**
- **last longer, to maximize return on investment.**

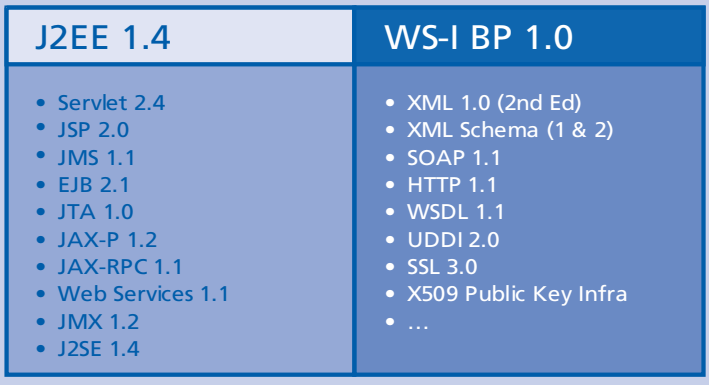
The Java Community has understood the principles underlying this approach and is building a technology 'stack' that closely matches the needs at each layer. The proper separation of concerns and information hiding are two key guiding principles. In J2EE specifications, for example, these principles have been implemented in the object models behind each of the application programming (APIs) and service provider interfaces (SPIs).

These same principles have also been implemented in the identification of key specific J2EE roles — product provider, application component provider, application assembler, deployer, systems-administrator, tool provider and systems component provider — and the skills requirement for discharging each set of related responsibilities. The goal in building this technology stack is to enable realization of modern layered architectures with maximal effect but minimal effort.

However this goal is only partially achieved in the current J2EE specifications. At the lowest layer, data is currently accessed through JDBC, Java Connector and JMS interfaces. These options recognize that data sources often already exist and must be re-used in the majority of implementation scenarios. Whilst the number and variety of data access components from vendors is expanding there is also an architectural intention in J2EE to rationalize and simplify this layer over time.

At the application layer the Java Community has focused on defining a relatively small number of components —

Figure 4.3: Standard checkpoint releases (proposed)



applets, servlets, JavaServer Pages, JavaBeans, Enterprise JavaBeans — that run in containers for scalability and ease of management. JavaServer Faces and Message-driven Beans have recently been added to this list.

However, the integration of XML APIs and processing into the Java libraries is having a dramatic impact at this layer as relevant XML standards mature. Further work is needed by the Java Community going forward to complete its integration.

The latest J2EE 1.4 specifications focus on the integration layer in the stack. The implementation of WSDL service interfaces, as a facade for underlying application components — whether programmed in Java or COBOL in legacy information systems — will enable deployment of service-oriented architectures.

J2EE 1.4 is intended to include first class support for:

- **programming services**
- **instantiating WSDL service interfaces**
- **bindings**
- **ports in Web and transaction containers.**

As noted earlier, it is intended that this support deliver Java systems that interoperate fully with other WS-I compliant systems. Much more work, however, is needed at this layer to deliver fully functional capability.

For example JAX-RPC — at the most recent 1.1 level — does not yet support asynchronous service communications or transport protocols other than SOAP over HTTP. Despite this, some open source projects and J2EE products (for example, WebSphere) have already made available support for services that use JMS messaging in advance of the underlying standards to provide this support.

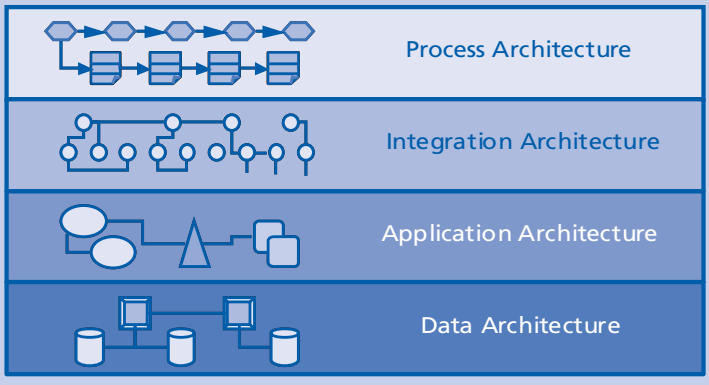
Equally, support for service request validation, authentication, authorization, confidentiality (encryption), non-repudiation and other security management capabilities have yet to be built into J2EE at the integration layer. WebSphere, for example, has included some of this capability in advance of the standards but more is needed to provide for fully secure service-oriented integration.

Process definitions

At the top of the architectural stack the focus is on abstractions for business process activities and tasks. J2EE does not yet include specifications for APIs or deployment artifacts that would facilitate mapping from abstraction to technology. The Java Community has, nevertheless, already formed expert groups focused on achieving this goal.

For example, the proposed 'Process Definition For Java' (JSR 207) specification will define metadata, interfaces and a runtime model that should enable business processes to be:

Figure 4.4: The architecture stack



- **implemented easily and rapidly using Java**
- **deployed in J2EE containers.**

This should provide a foundation to support tasks commonly encountered when programming business processes — for example parallel execution and asynchronous messaging. This should also apply for future Java implementations of business process standards initiatives, such as:

- **BPEL4WS**
- **WSCI**
- **W3C Choreography.**

Other process standard initiatives

The focus on Process Definition (JSR 207) and Business Integration (JSR 208) for Java systems is potentially the most challenging undertaken so far by the Java Community. At the same time there are:

- **research projects focused on business process rules and transaction models**
- **standards groups focused on refining XML grammars, such as WSBPEL at OASIS**
- **prototype technologies being developed by several vendors — from the likes of IBM, BEA, Microsoft and Collaxa**
- **the Java Community expert groups.**

This overlap clearly illustrates:

- **the development phases outlined earlier (in Figure 4.2)**
- **the rising level of complexity involved in producing open standard Java middleware.**

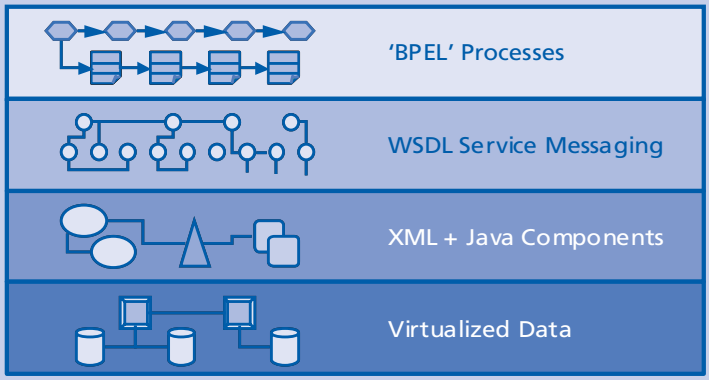
Outstanding issues

The Java System technology stack is clearly evolving at a pace dictated in large part by the independent standards organizations such as W3C and OASIS as well as the Java Community. Yet, there are several outstanding issues that must be addressed if the ambitious goals for J2EE are to be achieved.

The first is complexity. The volume of concepts, models, types, interfaces and other artifacts that are necessary to implement service-oriented business systems is constantly expanding. As the technology stack is built over time — from the bottom up — each layer should provide insulation through information hiding. In other words, each layer should hide the details of its underlying layers.

But this is not enough. The skills required to fulfill each of the roles identified in J2EE specifications must be managed to acceptable levels by smart tools that take on some of the burden. These tools in turn must be based on:

Figure 4.5: The technology stack



- **data about the designer's intention**
- **the implementation artifacts needed**
- **knowledge of the appropriate standards.**

The Java Community has taken reducing complexity as a challenge for the next generation of Java 2 platform specifications (expected in 2004 and beyond). Already expert groups have started work. The growing trend is recognized for annotating Java fields, methods, and classes with particular attribute values that indicate they should be processed in special ways by development tools, deployment tools, operations monitors or run-time libraries.

For example, the JavaBeans and EJB architectures have previously used naming conventions and deployment descriptors to encode information that could have been captured at source as metadata. The mapping between WSDL service descriptions and Java run-time objects is another potential use for metadata. The introduction of business process artifacts will be a third.

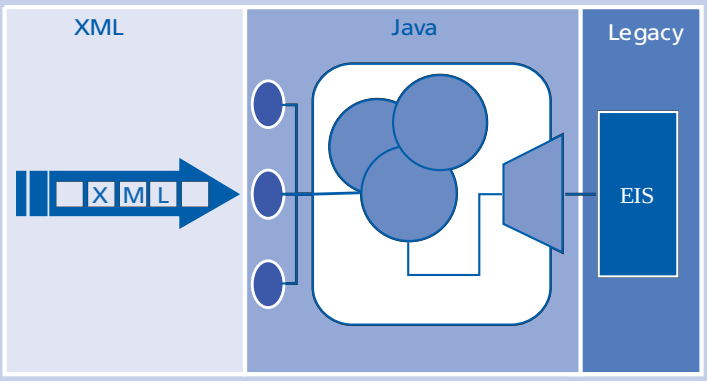
It is anticipated that once metadata has been introduced into the Java language and into the Java platform specifications, tool providers will deliver tools that greatly simplify the job of creating Java components at every level in the technology stack. XML will play an important role in capturing, transforming and exchanging metadata using these tools. Existing tool frameworks — such as Eclipse.org or NetBeans.org — will no doubt adapt easily over time to Java metadata, as they are already XML based.

The second outstanding issue (among possibly many more) is outlined in Figure 4.6. Increasingly XML grammars are being used to describe the contents of messages flowing between Java (and non-Java) platforms. Behind these grammars — such as SOAP, WSDL, UDDI and BPEL — are type models that are captured in XML schema. The issue concerns what happens to XML messages as they pass into Java containers and possibly on through adapters into legacy information systems (such as CICS or IMS or ClearPath) or into XML databases.

The correspondence between types defined by XML schemas and types defined in the Java language is being addressed by the JAX-B specification. With a few noteworthy exceptions it will be possible automatically to transform XML message contents into a set of Java objects (and vice versa) so that business logic may be applied in the normal way. This, combined with XML validation, should provide some level of type safety for messages participating in business processes.

However, XML deserialization into objects comes at a cost, and some industry standard business process documents are extremely large. It would seem foolish to transform large numbers of XML elements from a purchase order

Figure 4.6: XML to Java



document (for example) into Java objects just so that a customer credit rating can be validated as the first of possibly many separate steps in a business process.

In an ideal world, key elements would be extracted from a document as needed without the need to process the complete document. This is just one of the challenges of document-oriented service programming — a topic that is about to become extremely important to service-oriented open systems engineers.

Management conclusion

Java is not just about programming in objects, components, services or processes. It has evolved over the last eight years into an architecture for middleware for commercial systems that embraces four different types of platform:

- ***JavaCard for chips and embedded devices***
- ***J2ME for mobile and wireless equipment***
- ***J2SE for desktop and kiosk applications***
- ***J2EE for business application servers.***

Large numbers of each Java platform are being deployed in 2003 and there is no end in sight for this expansion in Java processing. Behind the success that Java is enjoying is a large community of developers focused on refining the language, the architecture and the applicability of Java to new problem domains.

Today the focus is on XML Web Service-oriented systems. Tomorrow the focus will turn to business process-oriented systems and tools. There is no doubt, at least in Dr. Jones' mind, that Java is growing and maturing as the community continually breaks new ground.

Implementing Web Services

Peter Bye
Consultant
Unisys Systems & Technology

Management introduction

Web Services have been extensively 'promoted' during the past couple of years or so. Perhaps promoted is too weak a word for the level of hype the industry has seen, as shown by the number of publications on the subject and conferences discussing it. Words of caution have come from various directions, not least from an analysis by Tom Welsh (Six things that could spoil the Web Services dream that appeared in the February, 2003 **MIDDLEWARESPECTRA** *(Volume 17 Report 1). Nevertheless, expectations continue to run high.*

But beyond all the exaggeration, and the attendant risk of disappointment, the technology of Web Services is likely to play a valuable role in the future of IT. Indeed it already is. This analysis by Peter Bye looks at ways of implementing Web Services, and illustrates with an example.

In his analysis, he considers:

- ***the context in which Web Services technology is being used, both in terms of the functions it is expected to provide and the environment in which it is to be implemented***
- ***the importance of basing implementations on a well-defined architecture***
- ***the approach used in the context of environments containing a specific system family, here the Unisys ClearPath OS2200***
- ***an example which shows how Web Services have been implemented in a public sector organization using ClearPath to deliver the application services.***

The Web Services context

As with most IT-related developments, Web Services are not a revolutionary or even a new concept. They are a logical extension of ideas that have been in circulation for a long time — distributed computing, assembling applications from components and so on. Building pragmatically on the experience of the past may be the best way to ensure success in the future.

There are, therefore, two aspects to consider:

- **what Web services are used for**
- **the IT environment in which they have to be implemented.**

Consider each in turn. One long-term vision of Web services is to provide the means for constructing applications, which are built from a number of what may be called 'service components', connected to the Internet. An end user would, for example, invoke a service resident in a system somewhere. That service would then invoke others, resident in other systems on the Internet, which would collaborate to deliver the required functions to the user.

Various standards, such as WSDL and UDDI, provide the means of advertising these services and then finding them at run time. SOAP provides a standard protocol for the invocation of such services (it may be carried by a variety of lower level protocols) — the standards do not mandate one — of which HTTP is the most common. In effect, this vision is a wide-area implementation of the long-standing idea of building applications from re-usable components that are connected at run time.

As has been pointed out by many people, great care has to be taken if the results are not to be disappointing, particularly in performance. These difficulties may be avoided if implementations are specific and restricted, certainly initially when experience is being gained.

One such specific approach is to provide the means for inter-system connection for e-business purposes. This could replace current EDI implementations with what many already see as a more standard, and common, technology base. The collaboration could even be confined to the members of a closed group, connected via an extranet. These restrictions remove some of the problems of scale, in that directories can be smaller because the number of providers of services is smaller. In simple cases, directory structures might be avoided altogether.

Another use would be to apply Web Services technology inside an organization connected to an Intranet. The purpose here would be for collaboration among intra-organization applications owned by that same organization. Again, depending on the scale of the requirement, it may be possible just to use SOAP and agreed XML-formatted messages for exchanges of information, thereby avoiding the need for directory structures requiring WSDL and UDDI. This approach is seen by many analysts, for example Gartner, as a good pointer to gain familiarity with Web Service technology.

One attraction is that the standards are widely available on many platforms, or at least committed to by their suppliers, thus providing a standard means of communication within a heterogeneous environment. An additional attraction is that, at least within a data center or campus environment, the network bandwidth available is sufficiently high to support the somewhat verbose structures of XML with decent performance.

As far as the IT environment is concerned, a major factor is the diversity of the systems present. Most large organizations possess and have to manage multiple environments, even in the rare cases where there is only one vendor. Having multiple vendors increases the degree of heterogeneity. In addition, when the applications used are of varying ages, they are likely to have been developed long before Web Services technology was available.

Factors to take into account

Therefore, those considering implementing Web Services for e-business and/or internal communication have to take various factors into account. How are these systems to be incorporated into a Web services environment?

As with any IT implementation, a systematic approach is necessary. The starting point should be the development of an architecture. This should represent the target environment. Within the framework provided by an appropriate architecture, technologies and products can be selected, and implemented in stages.

This sounds obvious, but all too often implementers resort to ad hoc, one-off solutions. The result, almost equally often, is increasing and eventually unmanageable complexity.

The role of an architecture

A fairly conventional architectural model, which is referred to as the middleware Bus, is shown in Figure 5.1. This is a somewhat simplified version of a model described in a previous analysis — *Middleware and distributed systems:*

some remarks on future challenges (**MIDDLEWARE-SPECTRA**: May, 2003).

In this model, various systems and access channels are thought of as being inter-linked by a software bus — the middleware bus. The various systems 'plug' into it for the purposes of inter-working, for the model permits two-tier implementations as well as three-tier ones, where a middle tier manages the access channel connections on one side and the various applications on the other.

The middleware Bus contains, at the center, what most refer to as 'core middleware' (again, see Figure 5.1). This is the basic 'plumbing' which connects the different elements together.

It can offer various connection possibilities, typically some form of synchronous procedure call as well as asynchronous options (such as message queuing). The attached systems can, therefore, communicate:

- **horizontally**
- **vertically, that is access channels to application systems.**

The applications, constructed from software components of various kinds (shown as the small circles labelled 'C') and depending on their environment, can be thought of as offering services accessed via the middleware bus. A user invoking a service would be connected to the appropriate application. The service might only require that application to be used or, if it is more complex, could require the application to invoke the services of others, either internally or externally.

Figure 5.1 shows two systems — A and B — connected to the Bus. System A is assumed to be able to support interfaces to the core middleware, referred to as the MBI (Middleware Bus Interface). If all the various systems connected to the middleware bus conform to the same standards, the result is something like a complete environment as offered by one of the middleware technology 'groupings', for example J2EE.

Not all systems will do this, however. Therefore the Bus has to offer other means of access. In Figure 5.1, System B is assumed not to implement the MBI. It is non-conformant and maybe using either some other form of middleware 'standard' or possibly no middleware 'standard' at all.

To permit access to the middleware Bus, the application can either be changed to conform or accessed via a gateway. This provides a bridge between:

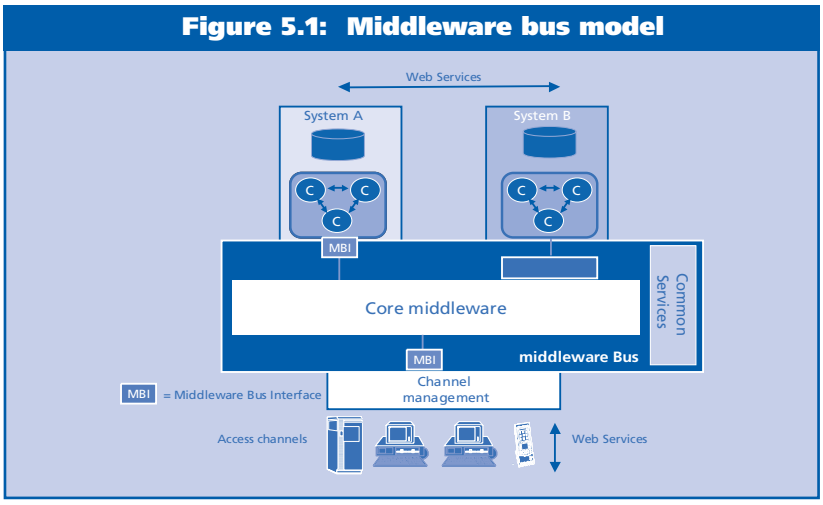
- **the MBI on one side**
- **the interfaces supported by the application(s) on the other.**

This could be as crude as screen scraping, or a sophisticated adaptation between two middleware architectures. J2EE, for example, defines various options for incorporating connections to other standards, such as JCA and JMS.

One other important element of the middleware Bus is a collection of what are called 'Common Services' (in Figure 5.1). These services include:

- **security functions**
- **message syntax**
- **semantic translations**
- **directories.**

Figure 5.1: Middleware bus model



Where do Web Services fit into this scheme? They can be viewed as access channels, adding to others already present, for example Web browsers or thick client workstations. They can also provide application to application connections, perhaps instead of or complementing or even using other technologies — such as message queuing. (The appropriate protocols obviously need to be supported in the middleware Bus for any of these to function).

Web Services and ClearPath OS2200 systems

Turning now to specific implementations,

this section illustrates how Web Services can be implemented in the context of a specific system family — the Unisys ClearPath OS2200 family. This description is by no means exhaustive, but it should provide at least a flavor of what has been achieved and how.

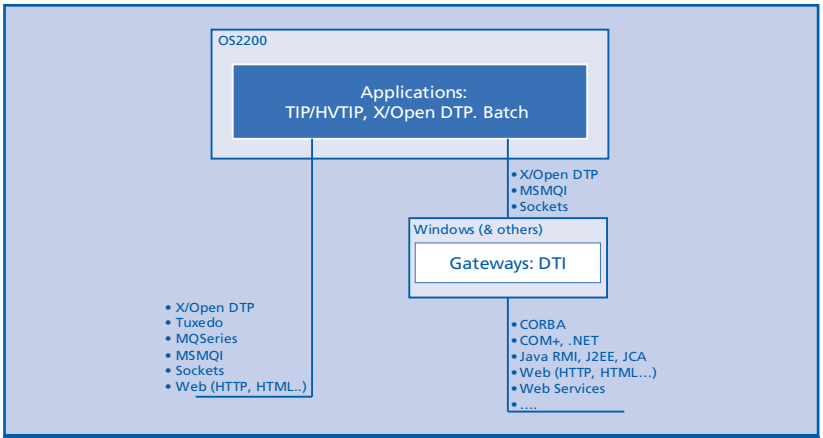


Figure 5.2: OS2200 middleware interfaces

The starting point is a brief outline of ClearPath systems (for those who are not familiar with them). Unisys announced ClearPath in 1996. One of the primary goals was to combine the OS2200 and MCP operating system environments — inherited from Sperry and Burroughs respectively — with other, off-the-shelf architectures in order to capitalize on widely available, software, middleware and applications running under Windows, UNIX and, latterly, Linux.

ClearPath systems support a variety of middleware. The focus is on enabling OS2200 and MCP applications to integrate with other systems using all widely accepted middleware standards. A selected set of critical middleware is implemented under the native OS2200 or MCP environments, with other, widely available middleware running under one of the other operating systems. (Not all will run under all operating systems: .NET for example must use Windows.)

A set of gateways, contained in a product called Distributed Transaction Integration (DTI), provides inter-working between other middleware and OS2200 applications. It uses a subset of the OS2200's native middleware. In the case of OS2200, the native middleware includes the following:

- **an implementation of the X/Open DTP model (now called Open Group DTP), which complements the existing TIP and HVIP transaction processing environments**
- **message queuing — both MQSeries and Microsoft's MQI**
- **a Web server environment, supporting HTTP/S, with a CGI interface as well as interfaces to TIP, HVTIP and X/Open DTP applications**
- **lower level interfaces, such as sockets.**

The natively supported middleware has evolved from time to time, as new environments have been added to native support or older ones have warranted being dropped. The above excludes 'non-middleware' interfaces — such as terminals, file transfer and so on.

The combination of these enables ClearPath systems to participate in most environments, both within an enterprise and between enterprises. The technologies able to co-operate with ClearPath systems, either directly through native middleware or through the DTI, include:

- **Tuxedo, as well as other implementations of the X/Open DTP standard**
- **J2EE, CORBA, and Microsoft COM and its subsequent evolutions, for example COM+ and .NET**
- **message queuing: MQSeries and Microsoft's Message Queuing (MSMQ)**
- **Web environments, either directly or in combination with other middleware**
- **JDBC for direct access to databases, including native ClearPath ones**
- **Web Services.**

Figure 5.2 summarizes the various options available with ClearPath OS2200 system, showing the direct OS2200 connections and those made through DTI. As can be seen, DTI uses a subset of the direct connections.

In principle, all the external connections shown in the Figure 5.2 (CORBA, COM+, etc.) can be mapped onto any of the three options shown for connection to the OS2200 environment — although there are some restrictions. For example, for the COM+ to OS2200 connection, support of global transactions with two phase commit between, say, SQL Server or Oracle in a Microsoft environment and an OS2200 database requires the use of the X/Open DTP connection. Many of the connections are symmetrical, in that either end — OS2200 or the remote ones — can initiate requests.

The system maps onto the middleware Bus architecture

shown in Figure 5.1. If the core middleware MBI supports one or more of the OS2200 native middleware architectures — for example, X/Open DTP or MQSeries — the OS2200 environment is compliant in that it supports the MBI. Otherwise, the core middleware can be accessed through the DTI.

From a deployment point of view, ClearPath OS2200 systems combine, in a single physical system, the OS2200 environment, together with a number of Intel-based environments, for example running Windows. The Intel-based software and middleware — such as the DTI — may run in the Intel partitions of the system or in free-standing hardware platforms, connected on a network.

As can be seen in Figure 5.2, Web Services:

- **may be implemented by using the tools available with, for example, .NET**
- **exploit the DTI to access the applications running in the OS2200 environment.**

Using the X/Open DTP implementation in the OS2200 applications provides the most flexible connection, as the X/Open model is inherently service oriented. Applications are constructed from services, which may be invoked by clients and may in turn invoke other services.

If this is the conceptual model, as seen through ClearPath, consider — in the following case study — how Web Services have been implemented in practice using the DTI and the X/Open DTP implementation.

A case study

This case study portrays a public sector (state government) authority. The authority concerned handles benefit payments, for example for food. It does this by issuing eligible recipients with a debit card. This card can be taken to participating outlets and, on check out, the card is swiped and a PIN entered so that the goods are paid for.

The cards are not made and delivered directly by the benefit authority to benefit recipients but by a contractor. The sequence of operation for the delivery of a new card is:

- **eligible recipients are identified by employees of the authority and entered into a database in a Unisys ClearPath OS2200 system, using various transactions**
- **a batch process runs regularly, extracting information from the database about requests for new cards**
- **this information is sent to the contractor making and issuing the cards**
- **the contractor responds, confirming a successful receipt or indicating an error**
- **the card is physically sent later, perhaps as much as a few days later, and a notification is sent back to the authority's ClearPath system informing the application that the card has been issued to the benefit recipient.**

The connection between the authority and the card issuing contractor had been implemented using SNA protocols, and had been in use for some years. A decision was then made to enhance the connection to use newer standards, in part because the contract with the card supplier was coming up for renewal and the invitation to propose needed to define a standard interface with the authority.

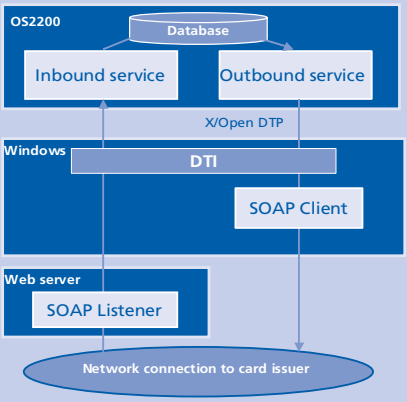
A number of options were explored, including:

- **FTP, with batches of requests**
- **message queuing (MQSeries)**
- **Web Services.**

The final decision was to use Web Services, as these seemed offer the necessary flexibility within a standards-based environment. They were also reasonable from a cost point of view. The new process therefore leaves steps 1 and 2 in place but uses Web Services for the remainder (rather than SNA).

The implementation was fairly straightforward and is shown in Figure 5.3. The sequence of operation is:

Figure 5.3: Web services implementation



- **eligible requests are entered into the database as before**
- **the sending batch program was modified into an X/Open DTP compliant client (shown as an 'Outbound service' in Figure 5.3) and this service extracts the data from the database and passes it to a new SOAP client, resident under Windows; the SOAP client was created using VisualStudio and the SOAP toolkit and the connection between the Outbound service and the SOAP client uses the DTI feature which provides a gateway between X/Open DTP and Microsoft COM+ (it is implemented as COM+ components, converting between COM+ on one side and X/Open DTP on the other)**
- **the SOAP client sends the request across a secure network to the card supplier, which possesses a SOAP listener and service to process the request**
- **after the card is produced, a SOAP confirmation message is returned to the authority**
- **a SOAP listener is installed in a Web server within the authority**
- **the SOAP listener passes the message to a DTI**
- **the DTI then passes the message to a new Inbound service, which updates the database as required; the Inbound service is an X/Open DTP compliant service.**

The implementation was and is straightforward, using the tools available to create the SOAP client and listener, and DTI to provide the mapping between COM+ and the X/Open DTP services. The advantage of X/Open DTP in this case is that it is service-oriented and is, therefore, easy to map to a Web Services request. This example also maps onto the middleware Bus:

- **the core middleware is COM+, with the DTI providing a gateway**
- **the SOAP connections to the card issuing company are part of channel management.**

A similar implementation could have been made using a J2EE application server, such as the WebLogic Server from BEA. The DTI would again be used, but with a different connection, using an EJB or RMI call from the Java system, among other options.

A few other comments about this implementation are appropriate here:

- **the same approach would apply to application system types other than Unisys ClearPath; for example, most systems support at least message queuing, which could provide an interface**
- **delivering a Web Service may require the co-operation of several systems within an organization (not just the one described in this case study); such systems can then co-operate using the common middleware Bus which may — or may not — use Web Services for internal co-operation.**

Management conclusion

In this analysis Mr. Bye discusses the implementation of Web Services. He stresses the need to be pragmatic when embarking on Web Services, focusing on what is achievable and building from there. He also emphasises the importance of working within the context of an architectural framework.

The fairly conventional middleware Bus explained here provides a scheme within which Web Services and other implementations can be constructed. This specifically addresses the common need to incorporate a variety of different systems using different architectures.

The Unisys ClearPath OS2200 system is a working example of a system containing a variety of critical applications, which may need to be exposed as Web Services. A set of key middleware interfaces into the OS2200 environment can be used directly from remote clients, or via Distributed Transaction Integration (DTI), which provides a set of gateways — thereby enabling additional client types to access applications.

The case study then illustrates a pragmatic example of where Web Services can be constructively deployed and connected to an application, running in the ClearPath system, which had initially been developed using SNA some years ago. It proved to be straightforward to add the Web Services interface using the tools available from Microsoft and Unisys.

Adopting a Service Oriented Architecture in a Web Services environment

Nick Denning
Chairman and Chief Technology Officer
Strategic Thought

Management introduction

Web Services are all the rage. Along with Grid technologies, to which they are closely allied, Web Services are likely to have a major impact on IT in the next five years. Therefore understanding what they might represent and how they might be used is an imperative.

In this context there are several generic benefits to deploying a Service Oriented Architecture which exploits Web Service technologies. This analysis, by Nick Denning — the Chairman and CTO of Strategic Thought (of Wimbledon, UK) — seeks to express some of the advantages and how IT might proceed.

What is a Service Oriented Architecture?

A Service Oriented Architecture is not a product. It is not even a specific implementation of Web Services technology (despite the common adoption of the word 'service'). It is predominantly a methodology. Indeed organizations have been building applications based on Service Oriented Architectures for many years (even though they might not have called them this).

Yet the history of the evolution of IT technologies has been a chequered one. The promise of each new technology is seldom immediately realized and often very large amounts of money are spent for little tangible benefit to the user. The current technology being advanced by the major vendors, such as IBM and Microsoft, is the Web Services infrastructure — using J2EE or .NET respectively.

The approach being proposed by both vendors embraces the concept of a Service Oriented Architecture. This is aimed at providing the framework within which the new systems using Web Services technologies can be designed, implemented and deployed.

Implementing Web Services

Historically technology churn has added significant cost and relatively little business value. There is a constant danger that adopting a Web Services architecture will follow the standard technology churn approach — and again add little value.

The biggest risk about this style of implementation (that we see at Strategic Thought) concerns developers trying to implement applications based on Web Services without due understanding of the implications. We fear that that they will:

- **construct Web Services as 'glorified' stored procedures**
- **force these services to rely on contextual information, which make them useful only under specific conditions within a particular application**
- **be too difficult, therefore, to use as generic, general purpose functions (as is the intention of Web Services).**

For example, consider some of the more obvious pitfalls ...

We are already seeing developers invoke logic as SOAP services when previously they would have used SQL or stored procedures. This typically breaks the transaction boundary, as logic is often executed in a series of services. Too often, we find, developers do not implement, or often even consider, the complexity of compensating transactions to back out work that is committed in early transactions.

The level of jargon that surrounds the discussion of service oriented architectures is also confusing. Developers using .NET client framework think that .NET means Web Services and then make the jump from .NET Web Services to a Services Oriented Architecture. Moreover, avoiding the construction of browser based applications by only using the forms-based .NET client framework to implement a VB like application, significantly reduces the benefits to be derived from Web Service technology.

In addition, we see applications constructed using lots of 'mini-services' with no thought to the design of the services interfaces themselves. These objects are services technically but they do not deliver a business service and cannot be managed independently of the client that calls them and cannot be used by multiple applications. Many of these services also rely on passing data that is specific to the current state of the client application. Again coupling too closely the client and the server means that the service cannot exist in isolation.

Over the years we have seen significant benefits arise from building applications that leverage stored procedures, because they reduce the level of TCP/IP traffic between client and server and the associated communications latency to a user screen as well as leverage query optimization within individual RDBMS products. Scrapping the RDBMS stored procedure concept — in favor of code running within an application server — may seem to be a retrograde step, but such an approach conforms more precisely to the purist's service oriented model.

This type of approach is, unfortunately, the opposite of what Web Services are intended to deliver. Yet we have seen examples of this type of development occur in traditional IT shops that have not thought through how to approach Web Services. In our view such narrowness of focus results in an abuse of the Web Service concept.

The benefits of implementing Web Services

At the same time, there are many Web Services standards under development and being delivered. The major vendors are developing early adopter implementations of those standards, which will inevitably change to some degree as each standard is ratified. We believe that it is vital to ensure that the implementation (within any given orga-

nization) is in accordance with defined standards, if only to minimize the risk of unnecessary complexity being introduced, or required functionality being ignored.

With those caveats to be borne in mind, there are various key benefits to using Web Services:

- **they provide (potentially) a direct vehicle whereby applications written in .NET and deployed to the desktop in a low cost environment can access back end enterprise services deployed in a J2EE environment**
- **implementing all Web Services (which follow a common architecture to enable them to be invoked by both SOAP (http and servlet) and JMS (WMQ and a message driven bean (MDB)) means that, by definition, each service must be atomic and cannot make assumptions about state other than that defined within the message passed, that the interface to the service will be under design control (and will not be evolved in an ad-hoc fashion as is the case where the individual programmer owns both client and server components) and that a service can make no assumptions about its caller.**

Furthermore, the work by the standards bodies to evolve the Web Services architecture is likely to produce a reliable and secure application framework. Even more attractive is that this should still maintain a considerable degree of diversity to implement a range of alternative services, thereby minimizing (or even avoiding) the historical problem of over-commitment to a particular architecture.

As such Web Services can provide:

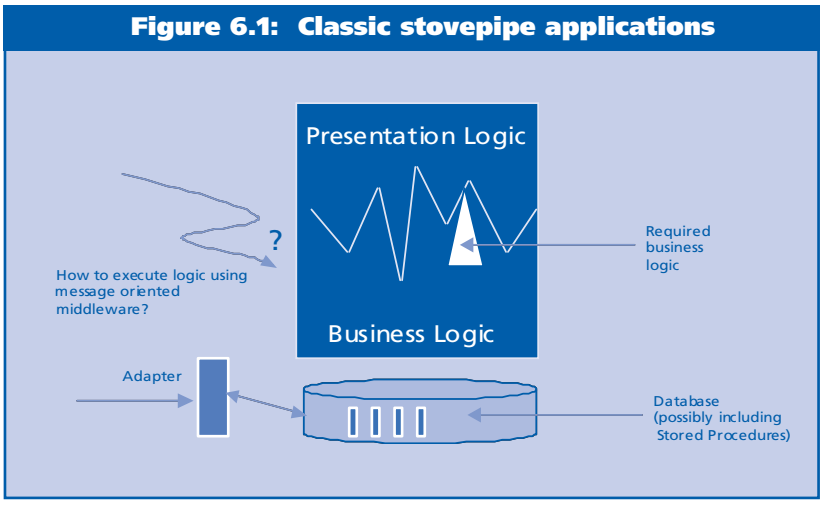
- **an effective baseline for development of a services oriented infrastructure (as described later)**
- **a potential infrastructure for implementing services (on an on-demand basis) between organizations — although, given the present state of development, it would be foolhardy (for reasons of security and transactional integrity) to implement cross business traffic over non-transactional SOAP; here AS2 or WebSphere MQ is essential.**

In a vendor context, where product vendors are able to open up their applications based on implementing a Web Services architecture, we strongly support this approach. However, we caution any customers proposing to build applications around a Web Services enabled product to seek assurances from the product vendor that the interfaces will remain stable between versions. If the objective is to build a new classic 'stovepipe' application (Figure 6.1) — based on a J2EE or .NET architecture, perhaps with some middleware for enabling an Enterprise Application Integration (EAI) architecture which interconnects applications via JMS — then the reality is that there is a limited requirement to deploy a Web Services architecture. Having accepted that, however, if an organization is:

- **prepared to invest in the effort to design — and maintain — those interfaces under configuration control**
- **amenable to evolving those interfaces to support multiple versions**
- **likely to want to develop a Service Oriented Architecture in the future**
- **wishes to ensure that services can be invoked via the emerging MDB JMS standard**

then there is benefit in investing in a Web Services architecture.

Figure 6.1: Classic stovepipe applications



Infrastructure benefits

An extremely important aspect underpinning Web Services technology is that so many of the facilities that would have traditionally been delivered as part of individual applications is now included within the Web Services application server technology. In the future we expect progressive development of commodity functionality, possibly available as open source code, that will significantly reduce the

amount of logic that must be written to deliver an enterprise, mission critical application.

Therefore a very important decision for any organization is the selection of the infrastructure, not least because the cost of training and adoption is non trivial (and should overshadow any product costs). Criteria that should influence selection include:

- **the financial stability of the supplier**
- **the degree to which application vendors are prepared to develop and deliver into the infrastructure**
- **the richness of the infrastructure currently**
- **a commitment to develop and extend the infrastructure.**

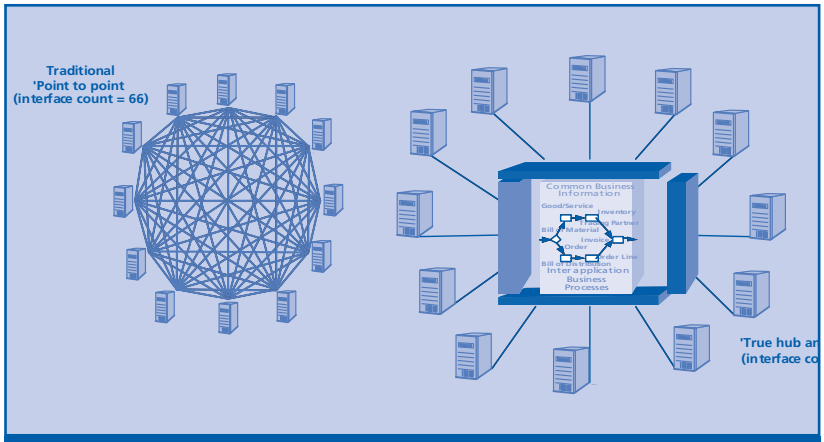


Figure 6.2: From mesh to hub and spoke

We currently believe that there will ultimately be two major players in the delivery of infrastructure software, IBM and Microsoft, plus the use of open source software plus probably one other survivor — from BEA, Sun or Oracle.

Service Oriented Architecture (SOA)

It is now appropriate to consider the characteristics of a Service Oriented Architecture and the issues associated with implementing such an Architecture. A key difference , that needs to be resolved, appears immediately. This concerns the way in which a Service Oriented Architecture is understood or described. Depending on one's point of view, a Service Oriented Architecture may be considered as any one of the three following 'approaches'.

A Service Oriented Architecture can be oriented towards building applications by invoking pre-defined services as if they were a set of enhanced stored procedures. There is relatively little change to the architecture of the applications being built, at least from a conceptual point of view. The Services to be built require additional design rigor — to ensure the linkage between any particular client and service is minimized so that each Service can be invoked by many applications. The key benefit here is being able to leverage the tooling and the middleware that can improve productivity and reduce the time to market.

Alternatively a Service Oriented Architecture can be considered as representing a substantially more fundamental change in approach. Historically stovepipe applications were connected using EAI techniques (Figure 6.2) to enable business processes to transition across the business through multiple applications. One alternative is to adopt an architecture for an application based on implementing a work flow that runs horizontally across the business. Where this application is built from a set of business services, these can be re-used to deliver new or enhanced work flow applications.

This model can be extended to adopt an approach that implements straight through processing (STP) with supporting exception handling. This matches more closely (than the first approach) the needs of a business participating in business procedures that span organizations where business processes must be connected.

In the first approach, the challenges in adopting a Service Oriented Architecture are relatively straightforward and basically involve extending the technology approach to building applications. There are some risks but these are:

- **relatively straightforward (well understood)**
- **practically the same as those previously faced by developers moving from the two-tier to three-tier application architectures in the past.**

The second SOA approach manifests a far more fundamental change. It is this that we, at Strategic Thought, consider to be:

- **the real challenge lies in designing and implementing a Service Oriented Architecture**
- **a true Service Oriented Architecture.**

Historically applications were stovepipes, with each developed separately. Even when common code was available, it generally lost its common code base when used on multiple projects: each project would take a separate copy of the

code, and make the necessary modifications to support the relevant operating system. In addition, additional enhancements (specific to each project) were almost invariably made.

The key benefits to be derived from a modern Service Oriented Architecture are that its resulting services will be:

- **built on a common code base**
- **implemented in a single language**
- **portable across all the platforms on which the service will run**
- **stored in a common repository.**

To us a Service Oriented Architecture must:

- **be implemented using a straight through processing model with exception handling (even if some services have exception handler business rules that result in 100% intervention with some services)**
- **include a transaction model that ensures that instances of a business process cannot be lost**
- **provide a compensating transaction mechanism to enable parts of business transactions that are already committed to be backed out in the event that a complete business process cannot complete.**

The approach that we, from our project experience at Strategic Thought, prefer is to rely on persistent JMS messages as the underlying transport layer. Logically we would make use of SOAP and a persistent state engine to note all transactions that are 'in play'. This would enable us to restart or back-out transactions, as required. But this is not yet available (due to the lack of suitable state engine support), although we believe it will be viable in the near future.

Message-based business process support

Message-based business process support brings with it a number of implications in an SOA context:

- **any Service Oriented Architecture should be based on a middle-out design that focuses on specifying the process to be executed, on defining the messages to be carried through the process invocation — and only then considers the client and server logic required to deliver the process**
- **the primary transport layer should be JMS or Websphere MQ which is used to provide transactional protection in the event of a failure so that transactions cannot be lost (as could happen if only SOAP/HTTP is supported)**
- **Services must be constructed so that the underlying business logic can be invoked by both a JMS invoked MDB or via a SOAP invoked servlet**
- **each Service will probably be made up from a transformation component, a validation component, an execution component and an exception handler to manage validation failures**
- **a compensating transaction must be able to back out the physical database transaction as a consequence of a subsequent service failure**
- **Services will be grouped into work flows where the characteristic of each is that it will comprise a number of Services which will either succeed or fail together (work flows will be either long running ones or short lived ones)**
- **work flows will be grouped into business processes where each work flow will have its own 'internal state engine' to manage the processing of the work flow appropriate to the individual work flow architecture**
- **each business process will be managed by a business process state engine that will invoke work flows and receive the completion status from the work flows, thereby determining the next work flow to be executed.**

Applications surrounding the Service Oriented Architecture will, therefore, fall into one of three types:

- **'work flow' or 'event driven' ones, where users are expected to respond to tasks as requested by the infrastructure**
- **'customer care' ones, where the user executes an application in which the user is in complete control of the work schedule being performed**
- **'dashboard application' ones, where data is broadcast by the underlying applications.**

In the context of the above, we expect that customer care and dashboard applications will represent a mixture of both types of functionality:

- **a customer care type application will have a number of dashboards that present data, and will be updated while the user is observing the screen (and from this screen a user will be able to invoke commands)**
- **a typical dashboard application will have a single dashboard screen, but from this a number of application operations can be invoked.**

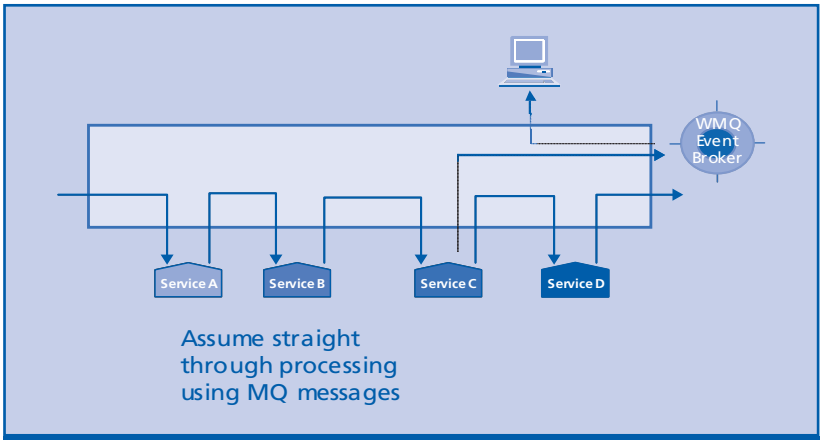


Figure 6.3: Straight Through Processing (STP)

The key issue here is that it is unacceptable to build applications that require applications to poll a database to maintain dynamic screens up to date. Instead, in our view, it is a requirement that screens which become out of date are updated automatically by a publish and subscribe architecture.

Another issue needing consideration is whether a Service is sufficiently 'right sized' to be able to implement a well defined business service. The degree of coupling with other Services must be defined but should not be expected to be large. That is, a relatively small number of Services should be offered against a single database or resource manager. Minimizing the number of permanent side-effects, that might affect other Services, is an area that should not be ignored.

This means that there should be a balance between having:

- **only one Service per database (with a complex message definition that implements a complex range of functionality depending on the structure of the message)**
- **a large number of separate Services each with unique message definitions that may all interact with each other and thereby require extensive testing.**

An SOA which works with a Web Services architecture should, therefore, focus on trying to deliver business functionality. It should not seek to expose a large number of individual methods.

SOA benefits

From a design perspective, a Service Oriented Architecture fundamentally enables new business processes to be choreographed by any business user. This is made possible by the assembly of collections of Web Services (taken from a list of already created Web Services) which can then support the intended business process. The point is that these Services are defined at a sufficiently high level of abstraction such that a business user only needs to verify that the message generated from one Service can be passed to the next Service in the business process. This capability significantly enhances the ability to assemble new applications within an appropriate architecture and tooling.

The use of messages between services leverages the value of asynchronous messaging. This can be exploited to protect transactions. In addition state can be added so that the execution of an overall business process can be managed.

An SOA that is based on Java and Web Services offers a way forwards which can be deployed across multiple devices and environments — specifically those which span J2EE, J2ME and JMS. This facilitates integration of systems on, and across, many platforms.

Similarly, using JMS/WebSphere MQ or JMS/WebSphere MQe facilitates transformation across multiple transports appropriate to each particular environment. In particular, existing legacy applications can be brought into play, which can be accessed via Websphere MQ.

Indeed, legacy services can be wrapped in a Web Service to enable such services to be introduced into this assembly architecture. In turn this enables all Services to be designed and implemented in isolation from other Services, which has the added benefit of simplifying testing — because Services can be invoked and tested easily by test harnesses operating independently of the applications that call them.

Finally, the adoption of a straight through processing model, when combined with publish and subscribe messaging, removes many of the major performance and scal-

ability problems encountered with legacy stovepipe applications. This is because the state information, that is required to identify a particular transaction, is part of the SOA/Web Services-based approach.

Implementations and SOA

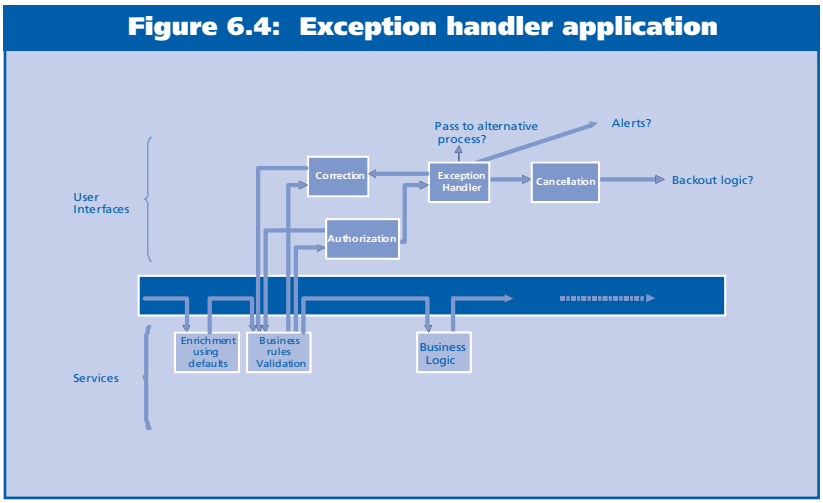
Various factors are important when implementing a Service Oriented Architecture. These include:

- **understanding that a Service Oriented Architecture should not be not dependent on Web Services; an SOA is a way of thinking that can be implemented using a number of technologies (for example we, at Strategic Thought, had implemented Service Oriented Architectures using a number of other technologies prior to the availability of Web Services)**
- **appreciating that it is vital to focus on a Service Oriented Architecture as delivering a straight through processing model (Figure 6.3) which uses a 'middle-out' approach based on specification of the messages between each service that will deliver the parts which will comprise the required business process**
- **ensuring that all user applications which will participate in a business process implemented within a Service Oriented Architecture exploit an exception handler application (Figure 6.4) using a work flow model (this will re-inject exceptions back into the system for re-validation once the exception has been handled)**
- **accepting that the management of state — both within a state machine controlling all processes and within individual messages to enable messages to be identified — is a vital component (this is not immediately self-obvious to many traditional developers)**
- **ensuring that implementations are undertaken in stages — to remove the 'fright factor' of trying to encompass too many technologies in one go (even though all the technologies are relatively simple); this increases the likelihood of quick and early successes**
- **structuring the initial phases to follow a goal architecture; this should minimize the need for any re-working if additional technologies have to be added into Services at a later stage (instead these should be selected based on minimizing the particular technology components that each phase requires)**
- **basing the standard implementation on Java — to provide portability — with code held in a central source code repository that is accessible to all projects wishing to use the common components.**

The process of evolution, once the early phases have successfully been delivered, must take into account the following requirements:

- **focusing on the testing infrastructure to enable effective, accurate and reliable regression testing**
- **ensuring that change management and configuration control have a high priority (to ensure that multiple versions of the Services can be supported)**
- **considering and planning carefully how to roll out new versions of all components of the infrastructure, and of applications, in a manner which allows continuous upgrading and avoids 'big bang' releases**
- **providing the tooling, and training, to business users which will empower them to choreograph new processes which can then be passed to IT staff for assembly and testing without incurring huge design, implementation or test phase costs**

Figure 6.4: Exception handler application



- **placing a high priority on the ability to monitor all parts of the infrastructure so that all aspects of 'the system' can be displayed and controlled; particular points for concentration include those where potential technical failures might occur — for example, throughput of individual transactions or groups of transactions plus detection of blocked transactions — so that these can be corrected and the system brought back on line and areas where business critical risks exist — for instance, work in progress which might present an unacceptable exposure to a particular client.**

Components needed to support an SOA

At Strategic Thought we believe that the following practical experience and software artefacts have significantly assisted us to support the specification and introduction of an SOA that exploits Web Services:

- **XML messaging, particularly in a publish and subscribe environment**
- **exception handling using work flow (and of work flow techniques, in general)**
- **knowledge of J2EE Application Servers (like WebSphere and others)**
- **compensating transaction techniques**
- **JMS, WebSphere MQ and Message Driven Bean techniques**
- **state machines, especially if implemented via databases**
- **application instrumentation**
- **monitoring and analysis, and recovery techniques**
- **security, at all levels**
- **testing frameworks and suites, as well building of test harnesses and regression testing**
- **risk assessment.**

While this list is not intended to be exhaustive, it should convey the message that SOAs when combined with Web Services do require depth and breadth to be successful. They are not simple artefacts casually to be drawn on the back of an envelope or pulled off a shelf and re-used ad nauseam. In our experience, each SOA is unique to each organization.

SOA dangers

That said, there are some dangers associated with SOAs. The primary one, in taking on a Service Oriented Architecture, is that it is regarded as a fad — without the long term encouragement being there to leverage the long term benefits that can be achieved.

The implementation, testing and deployment of systems based on an SOA requires a change in development methods when compared with the application stovepipe and EAI approaches. Organizations must be prepared to evolve their development methodology.

A secondary danger is to make the approach 'too big'. The approach needs to be owned by the program architect and implemented in an evolutionary series of small stages so that it can be progressively rolled out across an organization, and not via a big bang approach involving large numbers of staff. Only once the program and system architects have a well defined and proven standard should the SOA model be opened up to general scrutiny and discussion. Otherwise there is a real danger of a large technical IT debate developing that will consume staff cost and time in endless and unproductive talk.

What is missing? How to decide

Moving to a Service Oriented Architecture, and embracing the concept of business processes rather than stove pipe applications, requires changes in management approach. Probably the biggest challenge is not specifically IT-related, Instead it relates to the management of organizational change. Only with positive management can the benefits of the relevant technologies be delivered.

In this context, organizations using IT have various options. They can utilize standard products based on an underlying architecture about which they are largely unconcerned. This is the classic approach for small organizations with minimal development capability and little desire to build applications. Many such organizations implement their IT needs successfully by buying Windows-based applications based on Microsoft standard technology, bought from Microsoft and other third party providers. These organizations do not need SOAs, at least in the sense discussed above (Microsoft, in effect, has delivered it for them).

Then there are those organizations which have minimal or no development capability but which use specialized software that must run within an existing architecture. Typically such an architecture is dictated by the vendor of the software products. However the availability of browsers can now separates the infrastructure from specific applications. Where specific applications support all the business processes of the organization (perhaps with EDI to interconnect with external organizations and EAI for internal

connections) these organizations also have relatively little interest in an SOA (although their suppliers should do). Such organizations are assumed to have the skills to manage their infrastructure; in consequence they should mandate the use of an SOA approach and rely on their suppliers to deliver the necessary business process implementations.

Finally, there are those organizations where business processes depend on interconnecting a number of applications (probably originating from multiple sources). Where there is a need to change or enhance those processes on a regular basis, the adoption of a Service Oriented Architecture managed by the organization makes sense. A good test to determine whether organizations fall into this category is whether they have already adopted Enterprise Application Integration (EAI).

Our experience, at Strategic Thought, suggests that only those organizations that have already been through the pain of using EAI technologies to interconnect disparate applications will be able to appreciate the issues and understand the benefits of adopting a Service Oriented Architecture.

Management conclusion

Any organization planning to introduce a new technology stack, a new methodology or a fundamental change in development practices and procedures is taking on a non-trivial activity; a compelling case must, therefore, be made for major change. Recommendations for change must take into account all aspects of the current situation within an organization, the proposed goal architecture and the costs, timescales, phases and a number of detailed practical issues from which a business case can be made.

The key point about adopting a Service Oriented Architecture is that it is designed to promote an approach with regular, phased implementations that follow a strategy of migration and evolution. This approach must be followed from the beginning and, therefore, the initial deployment of the first component of the Service Oriented Architecture into production must be based on increments. This enables all those involved to be able to understand, manage and control the initial program of work to ensure it is low risk and meets the expectations of the business.

If achieved, the benefits of implementing a Service Oriented Architecture can be summarized as:

- ***improving the planning of IT spend***
- ***minimizing of the number of very large projects***
- ***reducing risk, by implementing projects in many smaller phases***
- ***enhancing the capability to change development schedules, to meet changing business needs***
- ***removing the requirement for large technology conversion projects (which provide minimal business benefit but do incur major costs)***
- ***leveraging existing IT assets more effectively (by wrapping them within a message based interface)***
- ***improving the quality of IT deliverables***
- ***responding faster to business needs***
- ***providing the ability to generate Return on Investment cases — providing the proof that the ROI target has (or has not) subsequently been achieved (thereby justifying further investment or curtail unproductive investment)***
- ***providing the framework within which multiple products can be integrated an the architecture and contracts can be let to develop sub-services (all within the well defined framework)***
- ***simplifying of staff training, combined with minimizing the number of staff required to manage the environment***
- ***enhancing business controls, especially the process for authorization of changes***
- ***providing an architecture that can be expanded on demand with minimum effort, thereby avoiding the requirement to purchase unnecessary capacity 'just in case'***
- ***improving IS, audit and control facilities***
- ***facilitating corporate events by enabling new organizations to be integrated into the existing architecture or alternatively the spinning off of subsidiaries with their own defined and working IT systems with the minimum impact on the parent organization's ongoing IT operations.***

While these benefits are not exclusive to a Service Oriented Architecture approach, an SOA makes a major contribution to obtaining these benefits (or the capability to deliver these benefits) in the future to a large organization that depends for its competitive advantage on effective and reliable IT systems. In the future we expect that the scope for organizations to adopt a Service Oriented Architecture will expand.

In the more distant future we anticipate the evolution of Service Oriented Architectures to be able to invoke external services on demand, but we do not anticipate this

approach being in regular use until an acceptable security infrastructure evolves and organizations begin to trust the use of externally provided, and paid for, services. For the time being, however, we anticipate the first take up to be to enhance in house IT architectures in a series of phased deliverables allowing organizations to provide the approach and evolve it to a full architecture implementation.

Can middleware replace the operator?

Mark Lillycrop
Principal
Arcati

Management introduction

Automation has proved to be an elusive target for IT operations over the last twenty years. Every step that the industry has taken to simplify the management of systems, networks, storage and peripheral sub-systems has been thwarted by additional complexity, as IT architectures continue to evolve and expand in unpredictable ways.

Middleware has an essential role to removing or masking predictable and repetitive system management tasks, potentially enabling operations management to focus on service delivery rather than infrastructural detail. In this analysis, Mark Lillycrop assesses this possibility.

Tackling the complexity

Rarely do IT departments obtain the chance to reduce the complexity of the environments they manage. Cross-platform technologies — including such developments as Web Services — promise to streamline the way that new applications (particularly those with an e-business flavor) are developed in the future. They offer the opportunity to 'front end' many existing legacy systems.

	2002	2005	2010
UNIX	45%	43%	42%
NT/W2	15%	32%	48%
S/390 + z/OS	40%	25%	10%

NB: this includes 45 times data center capacity growth between 2000 and 2010

Figure 7.1: Meta Group predictions

But, for the average large enterprise, the problems of drawing together and integrating numerous islands of computing, gathering consistent performance data from each platform, providing standard mechanisms for data access and translating this diverse computing environment into a fully-fledged 'service' (rather than a corporate overhead) remain unsolved. This carries a cost.

According to predictions recently published by the Meta Group (Figure 7.1), there will be a 45-fold increase in data center capacity between 2000 and 2010. As if that were not scary enough, this research suggested that the percentage of capacity attributable to UNIX, Windows and OS/390 respectively would change:

- **from 45% /15% /40% in 2000**
- **to 42% /48% /10% in 2010.**

Whether you agree with these precise predictions or not, there can be little doubt that the average data center will become more heterogeneous over the next few years. It is not that the mainframe is going away; far from it. It just is not growing as fast as its distributed counterparts.

Like it or not, UNIX-based systems are looking very attractive in the mainframe space. Linux is finding its way into the glass-house through Web Server scalability and blades. Microsoft also has its sights firmly set on the business-critical territory of the data center.

These diverse technical power-bases continue to restrict real integration within the IT environment. Furthermore, the antagonism between technical specialists on different platforms needs to be addressed head-on.

One solution is to consolidate server and storage platforms. This can certainly simplify the technical architecture. However, many CIOs find that the cultural issues involved in obtaining management buy-in for a major platform change are simply too difficult to overcome. Even when consolidation appears to offer substantial long-term cost savings, the short-term benefit of maintaining the status quo can be an equally powerful argument.

A less disruptive and more financially predictable solution is to use system management middleware to paper over the cracks between platforms. In essence this provides the benefits of consolidation but at a much lower level.

New challenges for IT

Of course, managing a multi-platform data center is but one of the complexity challenges that face large organizations today. Others include:

- **e-business performance**
- **mobile working**
- **scalability**
- **managing skills.**

Each of these is assessed below in the context of middleware and improving manageability.

E-business performance

The data center itself is relatively stable compared with the networked environment around it. E-business applications demand access to both front end Web-facing resources and back end corporate data from a diverse range of client devices and applications. Bandwidth is highly variable, and end to end performance is still almost impossible to measure outside the corporate intranet.

But within the e-commerce world, wherever a customer is located and whatever the time of day, there are still only three seconds to deliver the information needed or to respond to a transaction request. If the service falls below

The data center skills dilemma

For companies intending to retain IT data center services in-house, there will be a number of pressing skills issues to address over the next few years. First, the majority of mainframe operations and technical support professionals will be nearing retirement age over the next few years, yet there is a dearth of younger talent available to take their place.

The US-based Data Center Institute has recently launched an initiative to help companies retain older staff, while encouraging emerging IT and computer science graduates to take a greater interest in mainframe technologies and associated data center disciplines.

For example, Marist College in the USA, one of the most influential IT training institutions on the East Coast, is reintroducing an extensive range of core OS/390 operational course to its core curriculum, with widespread backing from US corporations that need the skills. It is not clear whether this problem will affect Europe to quite the same degree as America, but CIOs need to be prepared nevertheless.

Storage management is also heading for a manpower crisis. Storage is growing at such a rate that there will simply be too few administrators to go around. In his report The Future of the Business Network, Michael Peterson of Strategic Research Corporation said that, in 2000, the average storage administrator could manage 300 to 500 gigabytes. He predicted that, by 2005, the average administrator should be capable of managing about 4.5 terabytes, while the average site will have grown to over 50 terabytes.

Both of these issues will hit large IT users very hard in the near future. Without appropriate skills in-house, their only options are to outsource key data center systems — which is certainly not attractive in every case — or to provide improved automation tools and middleware to remove the more routine aspects of the jobs involved.

Figure 7.2: The skills dilemma

that magical response time, so we are told, the customer is liable to surf off in a different direction to a different supplier ...

The mobile worker

The internal employee is a much less predictable beast than he was only a few years ago. Teleworking is burgeoning. According to one source, nearly 30 million Americans (around 20% of the workforce) now work from home at least one day a week, and Europe is rapidly catching up.

More and more knowledge workers occupy a virtual, arguably unmanageable, office. The teleworker's PC is nearly always a portable device, packed full of company data but rarely subject to the same back-up, recovery or security scrutiny as local systems.

Wireless access to corporate data is proving to be a particularly thorny issue, with many employees establishing their own unofficial connections across the firewall. Many of these 'initiatives' breach company security policies in the process, whether wittingly or unwittingly.

Managing scalability

The Meta Group source cited earlier suggested a 45-fold growth in data centre capacity between 2000 and 2010. It could just as easily be 35 or 55-fold. Whatever, it is big, whichever way you look at it.

Even with the apparent slowdown of IT expenditure over the last couple of years, e-business moves relentlessly on. Every Web-based application that replaces a manual process, and every graphical application that replaces a text-based one, adds to the processing and storage capacity needed to deliver the required service, creating growth at every level of the IT architecture.

Skills management

One of the real problems with increased complexity is the demand it places on IT departments to employ armies of skilled support staff. For every different database, storage environment, networking standard and operating system used, a separate set of maintenance and support skills is needed.

This is where automation constantly struggles to succeed. For years, IT departments have been trying to provide tools that minimize the amount of human intervention required in day-to-day system operations, suppressing routine functions and using system intelligence to respond to the vast

majority of system-generated events.

Using automation middleware to relieve operations staff of routine tasks is essential for several reasons. With salaries and personnel-related costs accounting for an increasing proportion of the IT spend, head count reduction is more important than ever. What is more, IT could be heading for a skills crisis in many areas of enterprise systems management in the not too distant future (see Figure 7.2).

The shift from system management to service management

Clearly, with so many pressures on IT services, there is a serious need to re-consider the way that middleware is deployed to reduce the need for manual intervention in IT operations. There is little choice here — operations staff will ultimately become either too scarce or too expensive to make internal IT operations viable.

However, there is another influence on IT which is creating an even greater need to simplify and re-focus system management — return on investment. It is no longer acceptable to view IT as an investment in its own right. Instead we are rapidly moving into an era where IT is assessed only in terms of the service it can deliver to the organization.

If you talk to most IT Services managers — in medium-sized and large companies — they will say that their role has changed fundamentally in recent years. Instead of involving themselves in the capabilities of individual systems, they are now concerned simply with the overall level of service they deliver to user departments and customers. Furthermore, that is how their senior management wants to measure their performance.

As we have crossed the boundary from an infrastructure-focused approach to IT to a bottom-line service focus, the limitations of existing performance metrics and system management tools have become all too clear. In a heterogeneous environment, the way that different platforms perform, data is moved around and networks respond to variable traffic conditions make it extremely difficult to provide a single, unified approach to service management.

Many organizations have responded to this problem with a top-down approach, employing a single company-wide methodology for service management, and imposing it on every application and component within the IT infrastructure. One of the most successful ways of doing this has been through ITIL (the IT Infrastructure Library).

ITIL (www.itil.co.uk) is a set of highly detailed standards and procedures, developed by the UK government in the late 1980s and still managed by the Office of Government Commerce. The extensive ITIL documentation provides IT service managers with a set of best practices, which can be used to align every IT operation with the service level objectives of an organization as a whole.

The goal of ITIL is enormous, and in recent years it has expanded beyond its core focus of service delivery and service support to cover other areas such as:

- **infrastructure management**
- **application management**
- **security.**

Its popularity is booming. Membership of ITIL-based user groups (such as the itSMF) continues to grow fast.

Behind initiatives such as ITIL is the realization that operations and system management tools must be far better standardized than they are today — so that they can be used to set and monitor business-oriented service level agreements rather than reflecting the idiosyncrasies of the platform for which they have been optimized.

Vendor responses to 'IT Service Management'

Not surprisingly, the leading suppliers of tools for system, storage, network and data management have embraced the 'IT Service Management' principles wholeheartedly. Companies such as IBM's Tivoli, BMC, HP and Microsoft are repositioning their system management architectures to reflect the business objectives of their customers more closely.

The result is that we are beginning to see interfaces in management tools that enable them to be customized to specific business processes. In turn this enables IT departments to report performance levels in business terms, (although security still lags behind other areas of system management to some degree — as Figure 7.3 discusses).

A number of vendors are also battling with the inconsistencies of multiple platforms, in order to reduce drastically the number of management consoles needed to manage heterogeneous platforms. Projects such as BMC's Golden Gate are intended specifically to overcome the need for multiple database administrators, by:

- **interpreting the management information emanating from each database platform**

Security — the missing piece of the jigsaw?

One of the most difficult cross-platform system functions to automate is security. Security management is still a relatively immature activity, and the growth of its importance in recent years, mostly because of the added exposures posed by cybercrime, is out of all proportion to its sophistication. Platforms that have relatively bullet-proof security in terms of identity management and virus elimination are still struggling to come to terms with problems such as the automation of intrusion detection.

This is a challenge because security increasingly determines the level of trust that senior management place in IT. Security itself offers virtually no return on investment. On the other hand, neglecting it can have catastrophic costs and consequences for an organizations credibility with its business partners and customers.

One of the issues that system management vendors will need to consider, as they integrate more systems functions and incorporate them into middleware, is how to draw security management into the same process. The more intelligence that can be embedded in middleware, the easier it will be for irregularities in data and network access to be identified and dealt with automatically. As yet, though, security is still a highly fragmented and specialized discipline, and it will be some time before security applications can share information effectively with other management tools.

Figure 7.3: Security's jigsaw

- **translating this into a form understandable by a single, moderately skilled operator.**

Similarly, there is a huge amount of activity in the storage management arena, with companies such as EMC, Veritas and IBM finally automating the painstakingly arduous storage allocation activities that have made storage management the labour-intensive activity it is today.

Management conclusion

All of the activities discussed above are helping to reduce the overall operational headcount plus enabling staff to focus on higher level business objectives, while leaving more routine activities to be handled by system automation. However, for organizations to make real progress in aligning IT operation and business objectives, system functions need to pushed more deeply into middleware, thereby removing the need for users to spend time on detailed configuration.

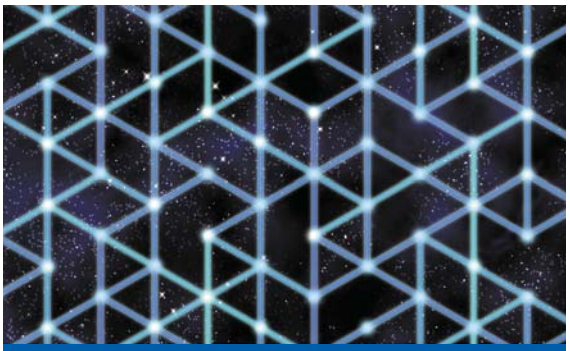
For this to happen, however, customer organizations need to see closer collaboration between vendors of system management products. The latter must build middleware-level standards for routine automation tasks.

Using middleware to simplify system management is a great objective. But, if the middleware itself is highly complex and subject to lengthy customization in every site, the benefits will likely be lost. The ultimate goals must be to:

- ***hide system management middleware from view wherever possible***
- ***concentrate scarce skills on the business-focused areas where the value will be measur-*****able.**

GRID.MIDDLEWARESPECTRA

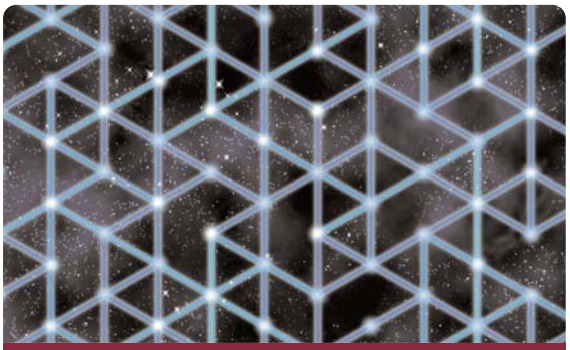
download from www.grid.middlewarespectra.com



GRID.MIDDLEWARESPECTRA

Applying the Grid to Commerce

1st Edition



GRID.MIDDLEWARESPECTRA

Commerce uses Grids

2nd Edition

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]