MIDDLEWARESPECTRA

incorporating FINANCIAL MIDDLEWARESPECTRA

Contents

May 2005

Volume 19 Report 2

Reflections on priorities at SAP Research

Burkhard Neidecker-Lutz
Technical Director, SAP Research
SAP AG

Management introduction

Burkhard Neidecker-Lutz is the Technical Director for SAP Research, which is the research arm of the SAP AG software group. SAP Research focuses on applied research, rather than basic research, and is primarily driven by two factors:

- ***the first has to do with challenges and questions that arise either from customers or from within SAP's own development operations; these tend to be relatively short term, with an eighteen month to three year horizon***
- ***the second is longer ranging, looking less at technological issues but rather at new business areas where SAP may not currently have an approach or offering as yet (perhaps because that particular field is not yet developed); one example of such a latter area was RFID technology, which six years ago was almost unknown.***

Mr. Neidecker-Lutz's responsibility within SAP Research is to co-ordinate the various research programs. At present there are seven of these, covering a broad variety of different areas:

- ***knowledge management***
- ***human computer interaction***
- ***business process modeling and management***
- ***smart items research (including RFID)***
- ***security and trust***
- ***enterprise services and semantics***

- ***software engineering and architecture.***

As not all of these are relevant to middleware, in this discussion Mr. Neidecker-Lutz focuses on:

- ***business process modeling and management***
- ***enterprise services and semantics***
- ***some elements of software engineering and architecture.***

Business process modeling

Work flow systems are relatively well established. I believe the basic technology — at least from a document perspective — is well understood and is relatively widely used in certain types of large organization.

What we find less often applied is business process modeling in smaller enterprises. Even though this is a large opportunity, it has not really been addressed by vendors. Furthermore, our research shows that there are relatively few, good functioning examples of cross or inter-organizational work flows.

My point is that there are a number of challenges here. Work flow, and/or business process modeling, and management currently are largely confined to those larger organizations which have the time and the resources to create a working system. But outside these, business process modeling and management often lack the tools and even the flexibility to deliver what I might call 'ad hoc work flows'.

This is an area where my research people are actively engaged. Instead of having just hard-coded work flows — which are notoriously difficult to adapt — we would like to switch to a model in which users can actually capture what we call 'out of system events', which can then be used as modifications of existing work flows.

If this is the most immediate concern, the larger issue concerning business process modeling and management is that it has an impact on the entire product portfolio of SAP, which is already undergoing a major change as it moves to adopt a Service Oriented Architecture (SOA) approach. This is not unique to SAP (Figure 1.1). We see a general industry trend — and not necessarily for technological reasons — which is taking vendors towards adopting Service Oriented Architectures for their products.

To me this is slightly weird. Is it a coincidence that — at roughly the same time — so many of the larger players, and even many of the smaller players, in the software market have informally agreed to jump on the SOA bandwagon?

The role of business process modeling in an SOA environment

The role of business process modeling and management, once you switch to an SOA environment, is much more important than it used to be in the past. In that past you had applications which had their own embedded process logic. Work flow would only be used occasionally, and only if it made sense as an external (to one or more applications) process co-ordinating capability.

In an SOA world, we are going to have services which respond to external stimuli. As such the only way you can have anything such as a process is to have some form of business process modeling and management capability to describe and then build the process flows that execute a given business activity. In this environment, management and execution must have the explicit function to drive a process through to completion.

My point is that work, or process, flow is changing:

- **from being an optional component that you might want to use**
- **to being one of the core elements of what you do when you introduce an SOA and related activities.**

The issues that we are researching at the moment are largely architectural. This is because the loads, the performance expectations and the kinds of interfacing issues that you will face when you start using something — that in essence is still traditional work flow — will force you to rethink simultaneously several intertwined issues. Adding more flexibility onto how you will implement means that many of the traditional ways of delivering simply will not scale up to the number of activities that need to be performed.

The Web Service dimension

Of course you can have an SOA without ever using a Web Service. We take a pragmatic view about this. On the one hand, I somewhat jokingly say that an RPC does not necessarily become any better just because you use angle brackets and a Web transport protocol. If you look at what happens in the lower levels — let us say at the less controversial Web Services standards — all we really have today is a glorified and slightly slower mechanism than we used to have in the past.

So, from a technological standpoint, Web Services and SOAs per se are not terribly interesting. What is interesting though — and somebody recently put this question to me in Brussels — is to consider 'why do you think that SOAs and Web Services suddenly will work and make a difference?' Such a question becomes even more relevant when one considers that we had CORBA and other not dissimilar approaches in the past, which usually did not become widely adopted. To me, what makes the difference is not the technological dimension.

What makes the difference is that, for possibly other related reasons or even strange coincidence, so many have agreed to move forward doing 'it' in a similar way. The good aspect about that is that this opens the possibility to start to tackle higher level questions — because we have increasing and near universal acceptance about the lower layers: we no longer have to worry about trying to agree on those lower levels. This does not necessarily buy anything concrete, except that it does become possible to concentrate on what are more interesting issues.

That said, this also introduces several artificial problems. Ironically, many of these are repeats of what we have experienced in the late 1980s when the apparent enthusiasm was for OSI standards. Many of the lower level mechanisms are good for either toy examples or medium message sizes. But these often break if you try to apply them unthinkingly to all scenarios.

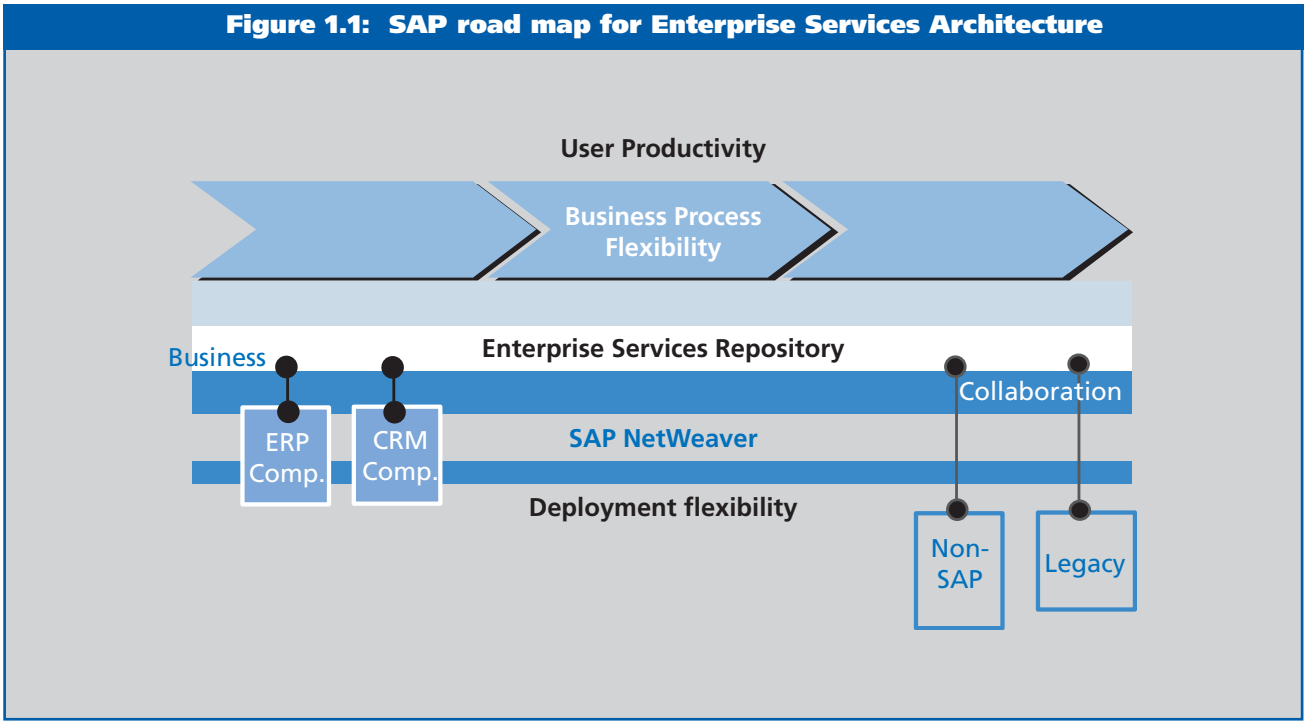
This simply does not work. Furthermore, it can apply both to scaling up or to scaling down. The lesson we have understood is that even if you are trying to do everything in a service oriented way, you must not do this unthinkingly — because, to repeat myself, it simply will not work.

This is, as you can imagine, a prime focus not only for SAP Research but also for SAP itself. Simply put, many of the problems that customers have concern data quality issues — not with IT quality but with the timeliness and accuracy of the information that flows in and out of any system (Figures 1.2-1.5). Eliminating as much of the manual dimension to data entry eliminates many subsequent problems.

So Service Oriented Architectures and Web Services are not necessarily related. But what we find productive is that we now have relatively well organized forums in which to discuss the various aspects of what needs to be done. You can even think of this as a form of 'aspect oriented programming' — which is being applied to finding the right terminology and mechanisms.

In this context I have working groups. But unlike the past, these are not artificially segregated into middleware, Web Services or application vendor or other specific groupings. Instead we have common forums for Web Services and standardization where we concentrate on the underlying challenges — and not on fighting the different horizontal segmentations or interests that we had in the past. That

Figure 1.1: SAP road map for Enterprise Services Architecture



said, this can also be confusing because we all suddenly have various levels trying to do everything to create a standard which might not technically work but is clear at the conceptual level. Indeed, I see a number of problems arising in the real world of implementation because the conceptual solution may not work so well in practice.

Business process modeling and management in the smaller enterprise

Another long term dimension which concerns us is that business process modeling and management (Figure 1.2) do not really relate to the smaller business or enterprise. There are reasons for this, which have lessons for us all.

One reason for the lack of acceptance applies to SAP just as much as to other large scale solution providers. This comes from the reality that trying to scale down solutions to fit the smaller enterprise does not work well.

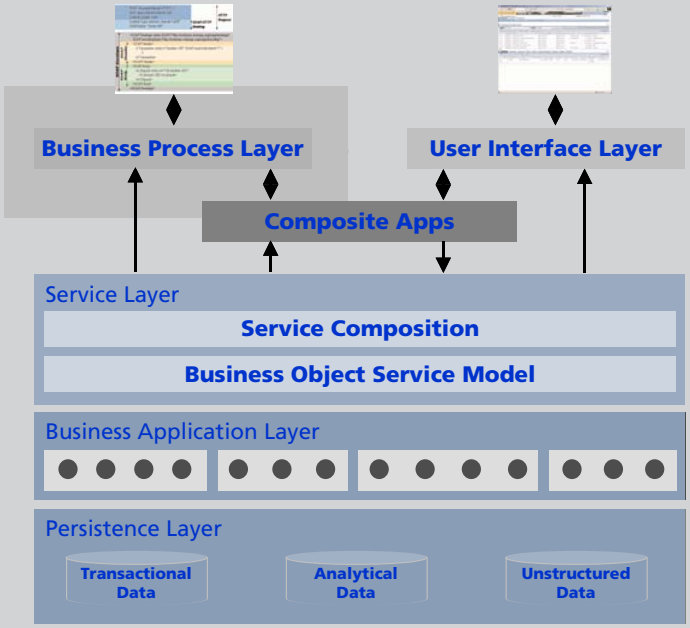
The reason is that large solutions tend to have much of their process logic embedded in their applications. Once you have products or applications that are structured like this, if you try to satisfy a market that has a different organizational granularity then the scaled down version that you are offering will likely not be appropriate for a smaller entity that may not possess the fine grained multi-role, multi-responsibility model that is embedded in most large solutions.

In effect the organizational sophistication appropriate to the larger organization becomes a problem for the smaller one, because the latter tends to have less formal structure and more informal responsibilities for individuals. The larger application solution now looks clumsy: it is not flexible enough for the smaller business.

Many of the steps that are explicit in a large enterprise work or process flow are implicit in the smaller enterprise environment — which is often the strength of the small and medium enterprises: they can react in more flexible ways without fearing to lose control. They have a higher element of human control and involvement available. Another way of thinking about this is that this is an instance where technology inhibits relevance to a different market.

That is one dimension. But there is another. One technological answer is to be able to de-couple the granularity of the process control flow from what is in the application. This is one of the drivers that explains why vendors like SAP are undertaking the tremendous engineering effort required to deliver products based on a Service Orientated Architecture. This investment will enable us to pull out the work and process flows as separate explicit components. Once we are there — which will probably still take us another two more years just to enable all the process flows across our own product set — we will have enabled much greater degrees of flexibility, as well as the ability to scale

Figure 1.2: Business process modeling and management



up and down. We have been working on this for three years already, and it may take us until 2007 to complete.

Inter-organizational process flows

This technological answer is partially also what will address inter-organizational issues as well as enable greater degrees of flexibility, not least for the smaller enterprise. Once you have flexibility about how you determine the way that the internal logic operates and how you can map this to external logic, you can tackle what typically inhibits inter-organizational processes.

The problem with cross-organizational flows — apart from all the low level issues involving authorization and security, roles, transport etc, which are difficult enough — is that you need a solid enterprise application integration infrastructure underneath. Fortunately there are several on the market today, including from SAP, although many competitive versions prefer to focus on lower level functions.

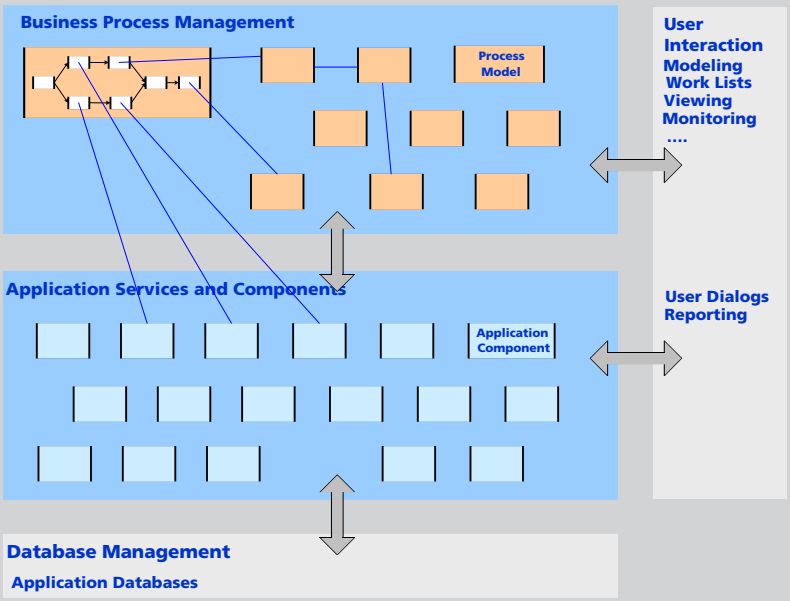
The problem really is one of trying to transcend the 'one-process model'. In an inter-organization environment there needs to be a model which will enable flows between all of the organizations involved. The model that was prevalent in the work flow products of the past tried to achieve a single transcending solution. But business realities dictate that you cannot hope to possess one global model — you cannot have one single process mapping between internal processes and external ones. For example, this is simply unrealistic from a change management standpoint.

Yet, businesses desire the ability to use multiple simultaneously active current models in flight, plus they need ways to translate not at the interface level but at a process logic level. That is part of the reason why you see an element of work flow in our NetWeaver Exchange Infrastructure.

Put another way, businesses need some form of instance matching between multiple process models in order to even begin to address inter-organizational issues. Arguably this capability will be needed more by smaller enterprises than by the bigger players (another reason why satisfying the small and medium enterprise is so important).

This is because most of the existing successful implementations of inter-organization work flows have occurred in supply chain management which have been driven (insisted upon) by single big players — the WalMarts or Fords or GMs of this world. So big are these buyers that they have been able to mandate that their own chosen approach be adopted by all those wishing to participate in their supply chain. That is fine and dandy only if you are large enough to be able to insist that you are at the center of the world — and can force such an approach via the application of draconian pressures.

Figure 1.3: BPM — a shifting paradigm



But for any small player that wants to survive without being tied either to any one customer, or to have to engineer for multiple customers with quite different approaches, it needs simultaneously to be able to service multiple process views of the world. In effect it needs to support a hub and spoke architecture in its internal system.

Ironically we are at a stage where we have the most complicated and adaptive infrastructure for our smallest customers for the future. That is slightly good, in the sense that we can take a little bit more leeway with performance, resource consumption, etc. in the technical solutions that we offer. This matters because there are a number of some slightly unnerving factors, not least being the amount of resource consumption that we face when implementing many of the lower level standards.

Enterprise services semantics

As I have already commented, there is a great deal of interest in Service Oriented Architecture — at least there are many, many vendor PowerPoint presentations with everything you need to know and describing how great and wonderful such an approach is. These are almost like holiday brochures, with the caveat that to obtain the benefits you must have understood all the implications and services in order for everything to be just fine one day.

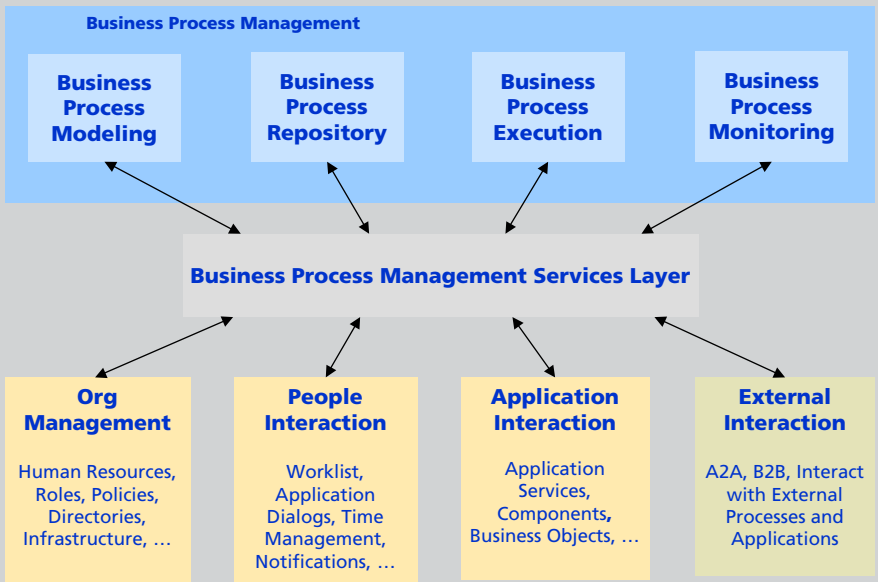
The short term question involves answering 'how does one obtain every application and service?' The challenge, and we at SAP know, is that a tremendous amount of engineering or re-engineering effort is required. If you are an application vendor such as ourselves and — depending on how you count — if you view every function that you support as (ultimately) a service, we in SAP possess 15,000 different services.

Re-doing every one of these does not make investment sense. Viewing every low level function as a service at an enterprise level — while we may do it at the Web Services level — risks losing sight of the objective. Something else is needed at the business object level. If we can do this we can reduce the number of services to about 1000 business services — and even this represents a huge task.

Much of our shorter term research revolves around exploring how we can re-engineer and assist the rebuilding of our existing applications without changing them (or breaking them) from the ground up while still service enabling everything we have. Partially this is about software engineering. But it is also about documenting the process logic in a machine readable form. It is not just about service enabling; it is also about discovering and then exposing a lot of the information that otherwise was hidden somewhere in documentation or repositories or, worse, was not documented at all.

If this is one part of our interest in enterprise service

Figure 1.4: BPM building blocks



semantics, the second half is associated with a dream — the desire that everything and all its associated components will magically plug and play together so that one can change suppliers, process models and other aspects all 'on the fly'. For businesses this is a reasonable objective. In IT terms it is necessary, if this is to work, to spend much time thinking about what flexibility actually means. This is another place that the semantics dimension arises.

Traditionally, as least at SAP, we have handled the integrating, putting things together and adapting them via the 'SAP eco-system' — third party organizations (like consultants and system integrators) that make their business from delivering integration and adaptation. Indeed, the revenue that arises goes to them, not to SAP.

If you now turn everything into a service integration exercise — because you no longer have any monolithic applications, just those 1000 business services — SAP would be dead in the water. Either it would alienate the eco-system by taking on the customizing or it would just be selling the services, which would need constant customizing which cannot be done manually economically.

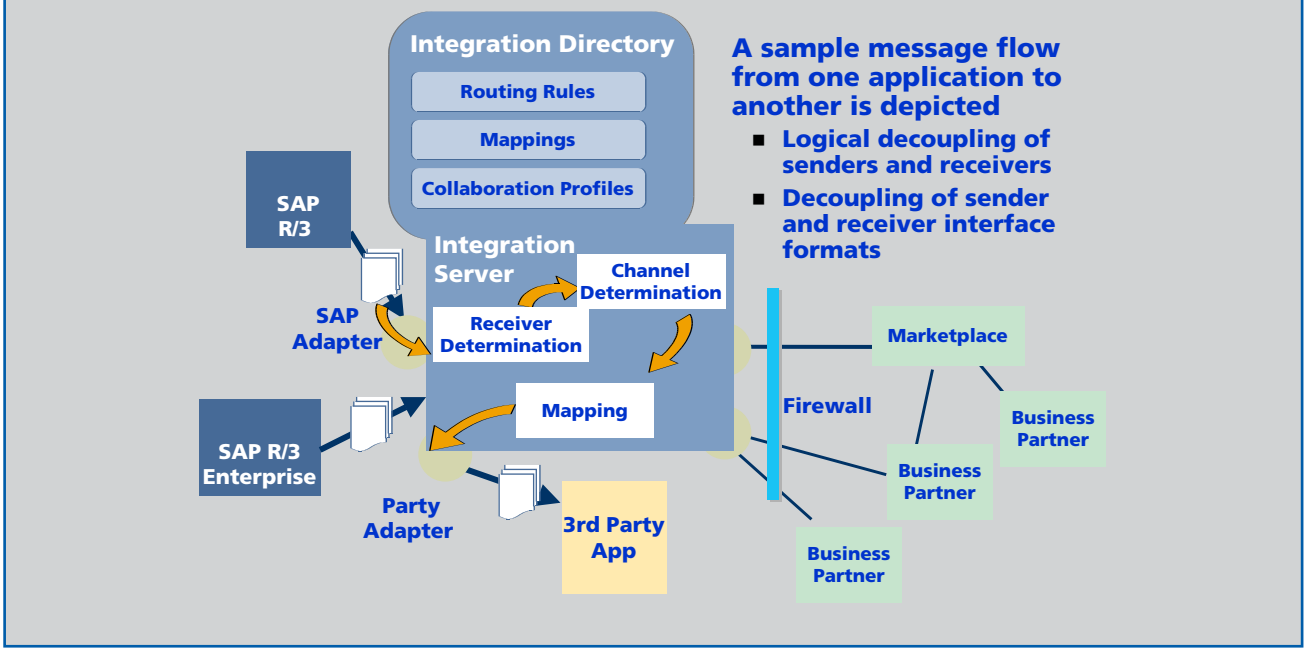
The underlying issue is how to semi-automate customization for each customer. The hope is — but, in the research community, we think it is a long shot — that somehow we can tackle the underlying problems using semantics applied at various levels. As a result, much of our research in this area is looking to see if it is possible to validate the application of semantics in this space.

Think a little bit about the 'Semantic Web', although we at SAP do not subscribe to this particular view of semantics. Nevertheless, explicitly capturing a lot of the know-how, the rules, etc. — all of which go into the various aspects of planning, design, interpretation and monitoring — is necessary. Semantics are involved at many different levels. Applying those intelligently to ease those problems — and ensuring that these work appropriately — is the longer range goal that we are trying to achieve in the enterprise service industry semantics space.

At present, such semantics are not yet a primary concern of customers — but they are also realizing this is an issue. When we look into the reality of how deployments work, an understanding of semantics is absolutely needed. In our own case at SAP we are not introducing the Service Oriented Architectures via one 'big bang' — because we do not think that would work. Instead we are carefully and gradually service-enabling more and more components. By doing this we maintain a functioning infrastructure while gradually introducing the service approach. Risk is minimized for customers.

The effect is that when a customer does decide to do

Figure 1.5: Decoupling of integrated applications



something new or different, the option will be there to use the new component and service capabilities. In this way organizations can gradually move to the implementation of services which do what is often partly pre-configured today. In effect you can start with what is new and so contain your risk. Over time the intention is that, rather than having to possess the complete semantic solution, you use as much as possible of what are pre-configured scenarios which can be adapted as you learn from these.

This is a cumulative building up, rather than just moving from everything being manual to switching to everything being automatic. Ideally this will allow you to train the people who were performing the manual tasks to operate in an environment which is increasingly automated. In such a way one can gradually replace functions.

That said we do not believe that semantic automation is going to be provided by some form of artificial intelligence type of approach, as might have been thought of 20+ years ago. What we are investigating is which parts of which technologies can be made to do something useful today that adds flexibility.

Software engineering and architecture

Our work on software engineering and architecture (Figure 1.6) is somewhat different from our other research programs in that our primary 'customer' is SAP itself — as a major software producing company. This means that the style of action and the topics are shaped rather differently.

As a software vendor the vast majority of what we sell is written in our own completely home-grown environment, specifically ABAP. This is something that has grown over the decades and has undergone considerable modernization over the years. As you might expect, there are good and bad sides to this.

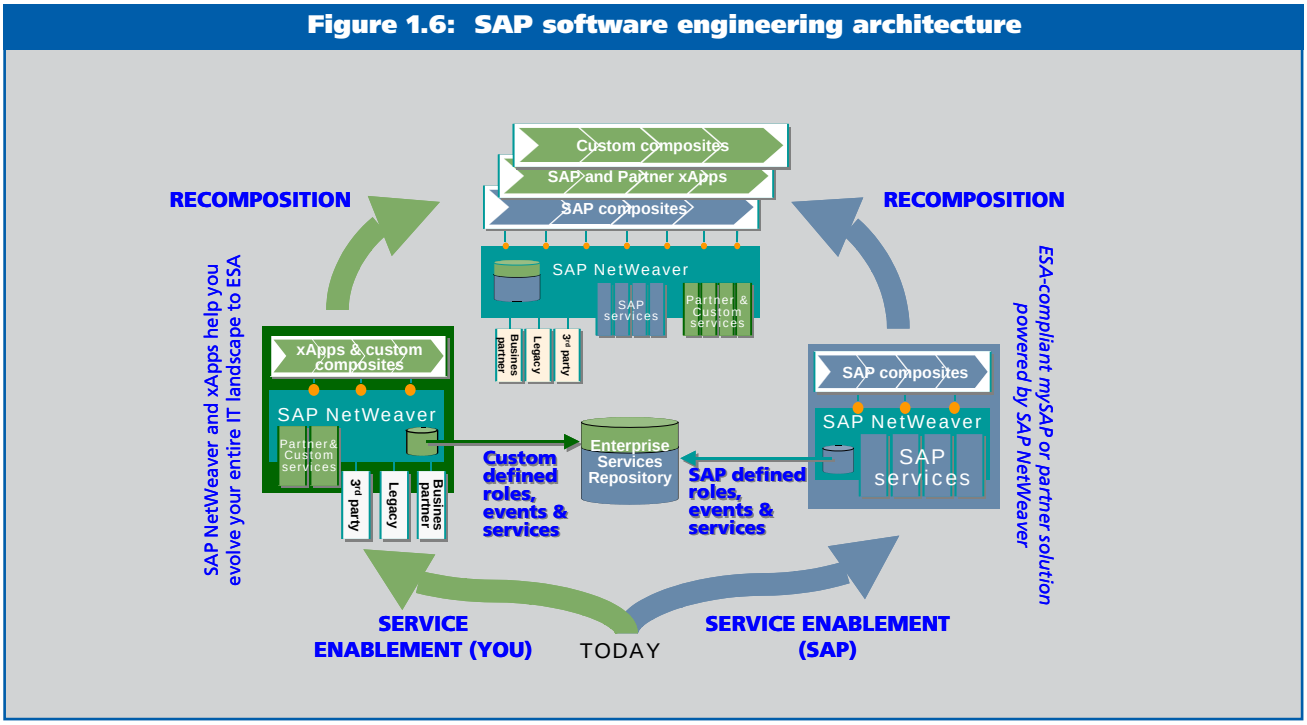
The good side is that we have complete end to end control over it. Another strength is that we have complete life-cycle management and deployment, etc. integrated into it right from the start.

One effort that has been on-going, and in which we at SAP Research have been heavily involved, is that a couple of years back the Company accepted the pressure from customers to include something that satisfies industry standards (rather than just having an SAP-unique approach). We added Java to our stack.

This has proven to be a mixed blessing — not just for us but also for our customers. Suddenly we realized what was already there in our existing environment — and therefore what was missing from the Java world.

Yes, Java is a more modern development language. But

Figure 1.6: SAP software engineering architecture



that does not mean it can do all that our own development environment offered. For example, life cycle management using Java raises questions about how to deal with change and how the subsequent results operate. The truth is that there are many aspects of Java that are really 'design-by committee' — and it shows.

As a result, one of our big shorter term efforts has been less about research and more about assisting the product group to bring all the development, deployment, management, de-bugging capabilities that we have in our own environment to customers who want to use Java in ways to which they are already accustomed in a pure SAP (for example, ABAP) environment. This continues to be a difficult challenge because we do not have the equivalent control (over Java). In addition there is a huge amount of infrastructure missing on the Java side if you want to provide practical life cycle management

Furthermore, we have had much research involvement in Java resource management and performance and debugging at run time. The practical problems that people see in the field are that Java application services are not as reliable for the scale of applications that we typically provide. For example, there is no simple answer to the problem of running out of memory if your application keeps growing or you have programming problems.

We do have various solutions to various of these problems at different levels and we continue to try to attack those problems. But, architecturally, this is difficult when you cannot 'do your own version' of Java. Sun does provide a separate version of Java for us but this is unique to SAP.

Model driven development and scalability

The longer term topic that we are trying to address in this space is model driven development. This is much deeper and wider than it might appear. You can even think of it as aspect oriented programming ... and more.

Our thought is that we would like to escape from all normal documentation. We wonder if it is possible to establish models for designing, developing, acquiring and even operating. As part of the software engineering we have been revisiting all the traditional layering and interfaces that we have.

What we find is that everything is tremendously stacked on top of one another to isolate layer upon layer of past dependencies. The long term issue that we are looking at is, while we drive towards 'model-driven everything', most of the old layering becomes irrelevant in the sense that you do not need to have all the old abstractions in place, some of which are 20 or more years old (if you think of the lower level interfaces to operating systems). To make matters worse, these are usually in some form of machine language targeted by some weird kind of compiler from a higher level.

The architectural questions we are asking are about what new abstraction layers can there be and how should we compress the many layers — that have been introduced over the years — in order to make the best use of where modern technology stands today. My point is that the degree of change is accelerating. This applies to hardware and software. There are new forms of hardware — multi-core, for example. Then there is more and more multi-threading: we need to know what to do about these to exploit them to the maximum.

At the level of systems, in the past we used to need to support twenty different operating systems and numerous database types. Today, the market dynamics essentially tell us that we need to support just two operating systems and one and a half processor architectures — for these now have 99% of the market.

This can open doors to opportunities. This is why we are looking at all the past layering and thinking about how we might obtain greater efficiencies.

Another area where we are focusing our attention concerns research into scalability and performance. This applies to both vertical and horizontal scalability applied to very large problems. This is also driven in part by the 'smart items' agenda — because the number of 'users' that a system might need to deal with will become so much bigger as more and more smart items are introduced. This can rise by several orders of magnitude in a short time frame.

A third area we are looking at carefully is how to make every single step to go faster. Adopting a service orientation means that latency becomes ever more of an issue. You have to design for this which further explains why we are making such a deep consideration of the layering. Inevitably there are physical limitations which will not be addressed by hardware — which means these must be taken care of at the architectural level.

Lessons learned

My first lesson is one that has been around for a long time — but it is still just as valid. This is, do not allow yourself to become hung up on the technology. Do not work from the

technology. Instead, understand what it is that customers are really trying to achieve. Only when you have understood this should you even begin to consider whether any technology is part of the solution. Deciding which technology to exploit comes last.

A second lesson learned comes more from the research perspective, and probably applies more to researchers than to the general business community. Technical skills are, maybe, 10% of what you need to get things done. Being able to mediate between people, to be able translate into conceptual terms, to be able to embrace and comprehend the viewpoints of multiple different communities — developers, customers, business managers or whatever — is what makes a research organization successful in the sense of actually being able to generate results. Having a brilliant idea but not being able to help mold it, or translate it into the world of the people that personally may need to use it, is of little use.

Related to this — do not forget that any idea needs to be sold and then maintained before being handed over to paying customers who actually need to try and use it. Failing to do this can produce a useless intellectual exercise or result.

Standardization has many positive aspects, as I have discussed. But what is often not seen is that there is a lot to standardization that few people understand. For example, I find that there are very few people these days who understand all the layers. You cannot even expect many people to look further than, say, one or two layers up or down from whatever is their specialist field.

This is part of the reason why we have an architecture research activity — so that we can place able people in positions where they can transcend this. But these people are extremely difficult to find. I know less about the curriculum outside Germany but within Germany it is now extremely difficult to find anybody with the right sort of training: our training has become very horizontal, and part of the reason is the tremendous trend towards layering and then taking layers as being sacrosanct, instead of occasionally posing questions about whether this is right for today.

Management conclusion

Mr. Neidecker-Lutz has a perspective which is uncommon. Not only does he work with the main development operation of SAP on the world's most commercially successful application but he has the opportunity to oversee key long term research about what we need in the future.

In this discussion he highlights some of the issues that SAP is addressing — and that customers will have to face in due course. In addition to his observations on SOAs, Web Services, enterprise semantics and model driven development, he is perhaps most interesting in his thoughts about inter-organization process flows and how these will need to be accessible as much to the small and medium sized enterprise as to large enterprises — if inter-organizational activities are successfully to be automated.

Progress on Grids and the e-Science Core Programme

Dr. Anne Trefethen
Deputy Director
e-Science Core Programme

Management introduction

Anne Trefethen is the Deputy Director of the UK government's e-Science Core Programme, which is responsible for supporting and providing the resources for e-Science application projects, and includes many Grid and middleware initiatives. The e-Science Programme was initiated in April 2001 with two strands of activity:

- ***the first was directed to the UK's Research Councils, to fund e-Science pilot applications***
- ***the second was directed to the Core Programme which is the part that has sought to solve generic problems associated with using Grids for e-Science as well as to establish and then support a Grid for the science applications.***

Dr. Trefethen obtained her PhD in Applied Mathematics before working for Thinking Machines. She then moved to Cornell University where she was Associate Director of the Theory Centre, one of the National Science Foundation supercomputing centers, at a time of the first developments of Grid technologies in the US. She returned to the UK in the late 1990s, before the e-Science Programme started and before taking up her current appointment, she worked as the Vice President for Research and Development at Numerical Algorithms Group which develops mathematical and visualization software.

In this discussion, Dr. Trefethen reviews what has occurred since April 2001 and the start of the e-Science Core Programme. As the Deputy Director she has been, and is, in a position to take a broad perspective of how Grids, and associated middleware, have evolved. She also discusses some of the longer-term middleware-related initiatives that have been needed and that are likely to continue into the future.

Setting the scene

The UK's e-Science Core Programme was charged with providing the infrastructure to support e-Science projects. These are projects which can benefit from the use of distributed resources — be that computers, databases, instruments or people.

To help build the knowledge base and deliver the required infrastructure, we set up nine regional centers across the UK (Figure 2.1). These have responsibilities to provide computer, visualization and data resources. They also act as information centers and we deliberately adopted this approach to attract external industry collaborations.

The UK's Department of Trade and Industry (DTI) invested GBP11M in this part of the Core Programme. Each of the Regional Centres received GBP1M and the National Centre — which, though it is located in Edinburgh, is directly managed by Edinburgh and Glasgow Universities — received GBP3M.

The net effect was to deliver a portfolio of 49 projects with 83 different groups collaborating and 61 different companies involved. In addition, more than GBP14M further investment was attracted from industry, with commitments of people to pre-agreed activities (rather than just receipts of machines or money). From the start we were determined that the Core Programme deliver true collaborations.

As for the industry participants, these are many — from the likes of major companies like IBM, Sun, SGI, HP and Oracle through to BT, Lloyds Bank and the BBC as well as 37 smaller players. The areas covered are equally diverse, for example from:

- **engineering**
- **pharmaceuticals**
- **communications**
- **the finance sector**
- **the environment.**

Results

These original industry-based projects started in 2001/2002. Many of them are now complete, and there are many success stories in terms of take up within industry of particular technologies — and with new products and services which have followed. These have included (using one illustration from each research Center):

- **Telemedicine (Cambridge) developed Multidisciplinary Team (MDT) meeting technology using ideas from the AccessGrid to allow clinicians in the West Anglia Cancer network to meet regularly to discuss patients; the technology allows the integration of microscopes and digital imaging technology so that clinicians can share information readily; although originally meant for a specific group within the cancer network it has become very successful and is used by a broadening set of disciplines with the National Health Service (NHS) investing in the technology on this trial network**
- **e-Diamond (Oxford) built a prototype system for creating a virtual federated database of digital mammography across hospitals in order to enable clinicians to access a wider range of cases for assistance in diagnosis, for the training of new radiologists and to give the capability to have scans considered by remote radiologists; this project has led to the GIMI project in the DTI Inter-Enterprise Computing programme.**
- **OGSA-DAI (Edinburgh) developed, in collaboration with IBM and Oracle, open standard middleware for the access and integration of data from distributed databases; its capabilities are already being used by projects in Europe, the US, China, Japan, Australia and elsewhere.**
- **G-Civil (Southampton) delivered a prototype Grid system for the collection, distribution and visualization of geotechnical data collected from civil engineering sites or from infrastructure monitoring schemes; it is now providing the basis for a service provided by Keynetix with potential 'uses' including the Channel Tunnel Rail link and Heathrow Terminals**
- **Computational Markets (London): working with a broad range of service developers and providers, this project has set about designing and implementing mechanisms for trading resources and services (these might include computer services, software, database interactions or network capacity) and these offer the basis of usage services for the National Grid Service (NGS)**
- **Gridcast (Belfast) has been collaborating with the BBC and the main aim of the project has been to develop techniques for broadcasters to share distributed media files and other distributed technical resources — by building on e-Science technology solutions; the infrastructure provides a quality of service in delivery and management of shared media across the BBC Nations, which has previously been lacking**

- **GridShed (Newcastle), together with BT, has been considering the issues of dynamic allocation and configuration of resources and routing of jobs and developing relevant middleware solutions; this initiative is tackling one of the major concerns of service providers (such as BT)**
- **RAVE (Cardiff) focused on visualization of data and is developing algorithms and technologies to permit collaborative visualization on a range of hardware (from large systems through to laptops, PDAs, etc.) across a range of physical network capacities; working closely with SGI offers immediate take up of the advances into the SGI product range**
- **BIOQuery (Manchester) focused on bioinfomatic e-Science technologies — and Bioquery is one example where an information integration environment has been developed, resulting in a system that provides a comprehensive and efficient query engine; if demand is sufficient, the two industrial partners — GeneticXchange and Sagitus Solutions — will seek to develop a commercial version of this.**

The e-Science pilot projects funded by the Research Councils are also beginning to show the benefits with new science results and research methodologies.

The second phase

The first phase of the e-Science Core Programme lasted for three years. A second phase of the Programme extended this to a five-year programme with additional funds coming from the UK's Office of Science and Technology. This second phase has taken the final overall investment in the e-Science Programme to a little over GBP230M, and runs through to April 2006.

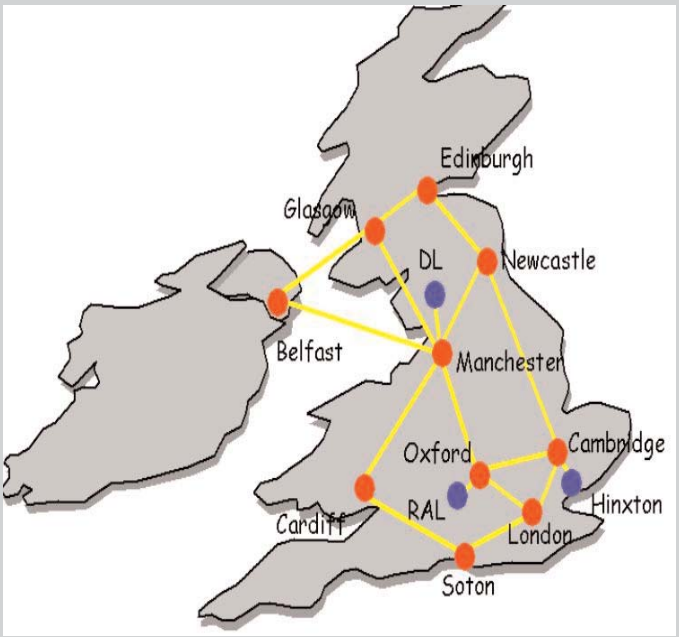
In the first phase we built what we called a "Level 2" Grid. This was an initial capability to link together resources at the various Regional Centres to enable users to learn to use resources in a Grid-enabled way. This was 'somewhat seamless' but was never as automated as will be necessary in the future. This was not surprising because assembling this Grid was itself part of the pilot or experimentation element of the Core Programme.

In the second phase we have tried to put in place more sustainable activities, to take those pilot activities forward in ways that can support the research community. This has involved extensions to what we had been doing.

NGS and GOSC

In the first phase we created a Grid Support Centre to support and assist development of the Level 2 grid and to assist users in understanding Grid software at the systems level.

Figure 2.1: e-Science Centres



That activity has now become part of our Grid Operations Support Centre (GOSC) which also supports our National Grid Service (Figure 2.2). In effect we have evolved the original Level 2 Grid that we had developed at the Regional Centres into a National Grid Service with specific resources. Today there are four clusters at different sites that have compute and data resources and these have been funded by JISC. The objective of the GOSC is to help users to understand:

- **what they need to know in order to be able to exploit the National Grid Service resources**
- **the different pieces of middleware that are available**
- **provide the security framework required.**

This National Grid Service is not finite. Part of its remit is to bring in resources from different places, different universities (including campus grids) to become part of the service. This requires the National Grid Service to define different levels of partnership.

I should, however, point out that this is not open to just anybody. To participate you have to possess what we call an 'e-Science Certificate', of which part is a security mechanism and can be obtained by anyone in an e-Science project or Grid related activity.

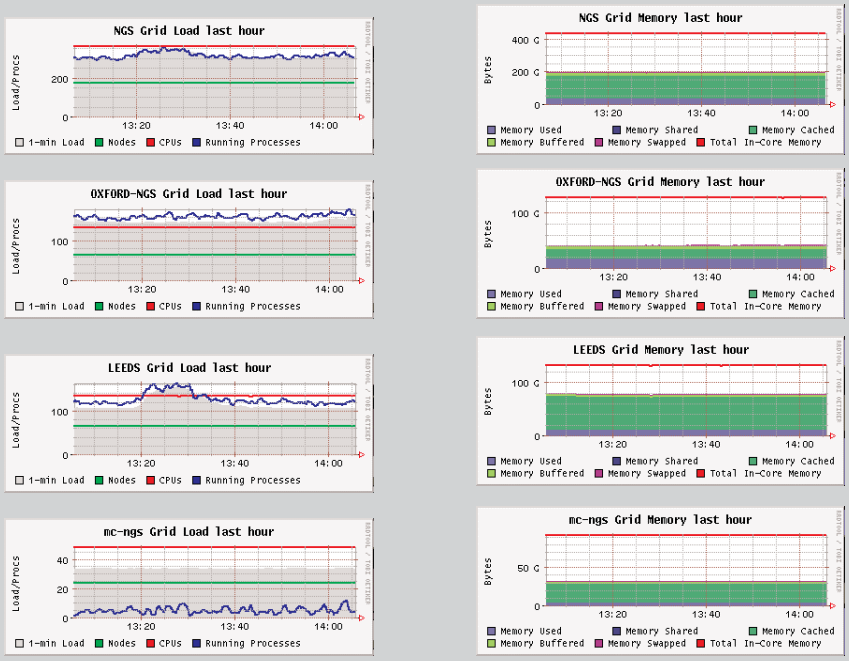
To use any of the e-Science resources you have to be authenticated with this e-Science Certificate. This is issued by our e-Science Certification Authority, that is run as part of the Grid Operation Support Centre. It supplies Certificates through pre-defined and agreed policies and mechanisms.

At the moment the National Grid Service's resources are freely available for anybody who has such an e-Science certificate. But at some point we will need to look at how the National Grid resources are allocated. In my view, the broader the user base the better the service will be. As the demand goes up we will need to look at how resources are allocated and which users will be able to use these resources.

At the moment we have about 200 groups — and rising — using the services of the National Grid. These include the usual number-crunching scientists — the chemists, the biochemists, the astrophysicists, etc. But what is interesting is that others are beginning to see the appeal. For example, we are seeing some social science projects coming through: they are using the National Grid for simulations and data federation.

Of course the sustainability needs to be more than continued provision of these services, and also needs to provide sustainability of the software developed by the projects.

Figure 2.2: National Grid Service



15

For this purpose we have created an Open Middleware Infrastructure Institute (OMII).

OMII

The OMII (Figure 2.3) has the remit to provide the middleware and services that can then sustain a Grid infrastructure for scientific research. The people participating in this are identifying and then taking, from the research projects that began in 2001, those generic components that have been developed and then:

- **making these more robust**
- **ensuring that the results conform to standards, as these are being defined**
- **ensuring that the resulting middleware is interoperable.**

This activity has now been active for 18 months or so. There has been a release of OMII Version 1, which is a first set of services that provide the basic building block for a research Grid. In the next few months there will be additional releases of services, most of which will have originated in the projects that were funded early on. These include such basic services as those for job submission as well as those required for (say) interfacing to engineering environments.

So, the OMII's actual role is to fund the building of robust Grid middleware from pilot or prototype project middleware. In so doing this will provide additional support to the other activities of the whole e-Science Programme. Part of this activity involves identifying gaps and then filling these appropriately so that the end result will be a full set of Grid middleware applicable to the support of e-Science.

I am sometimes asked about whether this duplicates what is already offered by the commercial software market. In places it may. But our experience is that commercial software vendors do not supply everything that scientists need, in part because scientists are not their main market. That said, one of the objectives of the OMII is that its services are developed as open source implementations while another is that such open source solutions be capable of working with products supplied by commercial vendors.

Another aspect of the open source impetus is to ensure sustainability. From inception part of the Core Programme responsibility is to ensure that what has been researched and developed is not lost, or has to be re-invented at some future time.

Equally, because we have adopted Open Source principles, these activities are international in scope. The OMII team are working with activities across Europe, in the US and in China. Our hope is that there will similar types of institutes

Figure 2.3: OMII

GRIMOIRES: Grid Registry with Metadata Oriented Interface: Robustness, Efficiency, Security	**University of Southampton, Southampton, UK**	A secure robust registry, built upon the UDDI specifications, that enables additional metadata to be associated with a registry entry.
WS-JDML – Web Service for Job Submission and Monitoring	**Imperial College London, UK**	The ability to initiate, monitor and control a job on a compute resource through a web service by using the emerging Job Submission Description Language (JSDL) from the GGF.
OMII-BPEL (Business Process Execution Language)	**University College, London, UK**	Through an open source BPEL implementation provide an environment to compose and execute scientific workflows across distributed resources.
FIRMS: Federation and Implementation of Reliable Messaging Specifications for Web Services	**Indiana University, Bloomington, IN, USA**	Will provide a reliable messaging infrastructure through an open source implementation of the WS-Reliable Messaging and the WS-Reliability specifications.
FINS: Federation and Implementation of notification Specifications for Web Services	**Indiana University, Bloomington, IN, USA**	Will provide an open source notification infrastructure based on the WSNotification and the WS-Eventing specifications.
Robust Application Hosting under WSRF::Lite	**University of Manchester, Manchester, UK**	Development of the PERL based WSRF::Lite environment .
Geodise Toolkits	**University of Southampton, Southampton, UK**	Adapt the current Geodise Computational, Data and XML toolboxes to interact with OMII services and from within both Matlab and Pyton environments.
OGSA-DAI	**National e-Science Centre, Edinburgh, UK**	Continual development and integration of the OGSA-DAI services into the OMII distribution.
WSeSS: Web Services for eScience through SSH	**University of Southampton, Southampton, UK**	The use of SSH as a means of connecting the Job and Data services in the OMII distribution to computing resource, and the exploration of failure in job workflows using BPEL .

established around the globe that can then work together although at the moment collaborations tend to be associated with major Grid projects.

Digital Curation Centre

Another institute that we set up under the second phase is the Digital Curation Centre (DCC). Again this was established jointly with JISC, and was created because we need to have some understanding of how to deal with all the data that is being created in the various e-Science — and elsewhere — activities.

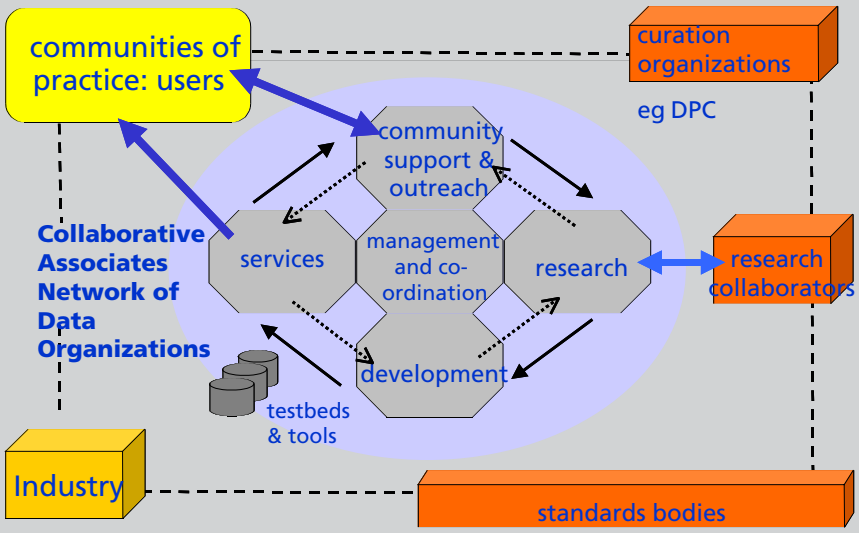
What makes this so complex is that computing — as well as specific research technologies — enable huge amounts of data to be created. The classic example is the petabyte, or more, of data that the Large Hadron Collider at CERN will generate each year. But the same applies to any other discipline, be that astronomy or bioinfomatics or whatever. In addition, different disciplines possess different pieces of knowledge as well as having different ideas about how to store this.

If you just look at the many science areas, we are already generating huge amounts of data — either through sensor technology or through simulations or through the creation of new databases (for example in bioinfomatics). The management of that data is something that, in too many cases, people have not thought through. How do you curate data in such a way that there is meta-data that describes everything you need to know about the data in order to make it useful? If the data is collected in an Excel spreadsheet, what do you need to do to make sure that, in 10 years time, you are able to look at that data and know what it represents and where it came from (Figure 2.4)?

Besides the curation issues, there are those associated with the management of large quantities of data and being able to sustain that data for the future so that it can continue to be useful. Then there is the problem of tracking data. Think about the implications of when a new piece of data is generated by bringing together other data from other sources: for that new data to be reliable it will need to have some record of its own provenance, plus the provenance of the source data. In some ways this very similar to the problems of tracking what is in food, and where did the ingredients that make up that food come from. It is also an important aspect for protection of IPR — who 'owns' the result of scientific research that brings together disparate data sources.

Yet another dimension concerns longevity and re-usability. It is quite possible that a researcher will want to use data in ten years time that was collected in 2005. Will he or she have the capability to access it and use it, and know sufficient about its provenance to be able to rely on the source to draw reliable conclusions? Also, what happens if a

Figure 2.4: Digital curation



researcher wishes to bridge different scientific disciplines that today store their research data, and results, in incompatible ways?

The Digital Creation Centre, which is based in Edinburgh, is looking to bridge and bring together the knowledge from many different sources, including (for example) the social scientists and digital libraries as well as the scientific domains. It has two tracks to its activities:

- **a research track**
- **a service track.**

Its remit is not to curate data. It is to provide insights and information about best practices.

The industry angle

The DTI was one of the initial collaborators in the e-Science Core Programme, jointly funding it with the Office of Science and Technology (OST). From the start we deliberately kept the two parts of the Programme tightly integrated.

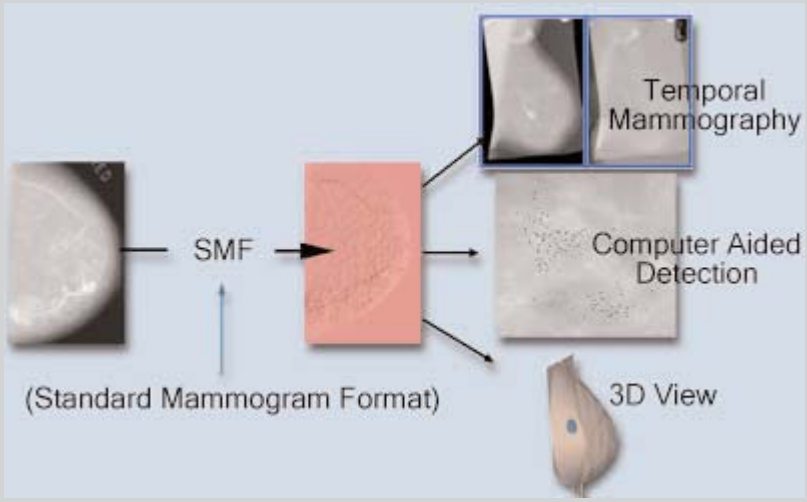
In the second phase the e-Science funding came directly from the OST because the DTI had moved to a different funding model — the Technology Programme. But one of the original DTI technology areas in this new programme was Inter-Enterprise Computing (IEC) — essentially focusing on the commercial side of e-Science and looking at how Grid technology could be introduced into industry and exploit the research that was being undertaken.

The DTI has now funded seven projects in this Inter-Enterprise Computing area (Figure 2.5), many of which have direct links to the projects and work carried out in the e-Science Programme. As noted above, the GIMI (the Generic infrastructure for Medical Infomatics) project, which is led out of Oxford University, uses as building blocks the technologies developed in the e-Diamond research project which extended digital mammography access across hospital sites. Another one is called Broaden and is headed by Rolls-Royce. This builds on the e-Science project called DAME that looked at analyzing the health of Rolls-Royce aircraft engines in flight. A third is CRISP (Commercial R3 IEC Service Provision). Led by BT this builds on projects developed at the Newcastle e-Science Centre regarding dynamic virtual organizations.

These are just three examples of industry putting into production some of the technologies that had been prototypes in the research projects undertaken and managed by the UK's Research Councils. While this is not tightly co-ordinated with the e-Science programme, what is important is that it is the same community that is participating.

In addition, there are two knowledge transfer networks.

Figure 2.5: eDiamond to GIMI



The first, IECNet, is a consortium between the National e-Science Centre and Intellect. It provides a direct link to and from the e-Science community for sharing what has been discovered. The second one is called GridUK; this is a smaller effort but aimed specifically at delivering information about working Grids to the commercial environment.

International dimensions

An area that I have not yet discussed, which relates to my previous points, is that we have consistently focused on making sure that we have international collaboration at all levels. This is for a number of reasons.

What we are trying to do in building collaborative infrastructures has to be done globally. There is no point in the UK inventing something that will not work with what others adopt as standards. But, equally, we need to work in the creation of those standards. From inception we have included in the e-Science Core Programme a number of mechanisms to ensure that we have the appropriate people working on standards, including funding people to participate in standards bodies as part of the standardization process.

Furthermore, we have also funded what we have called 'sister projects'. Where there is an activity in the UK that has either complementary activities internationally, or schemes that are very similar, we have funded a sister project to enable the participants to spend time working together, to have workshops, so that they can leverage developments on an international basis.

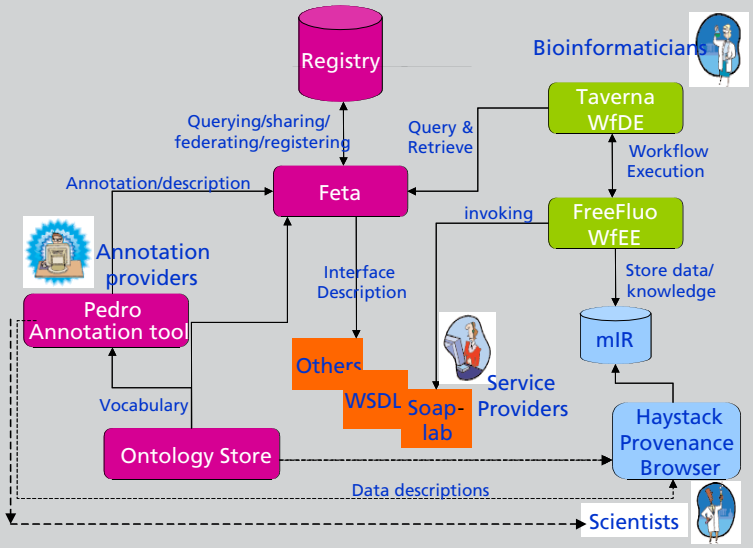
This continues to be very important to us. We have signed a number of agreements with different organizations to ensure that we are in lock step with what is happening.

That said, I think that we have been fortunate in the e-Science Programme in that we possess a tightly co-ordinated programme that is capable of bringing together applications and infrastructure development. This may sound trite. It is not. It has meant that the UK Grid infrastructure development has been driven by applications more than it might have been otherwise.

It has also allowed the UK to build an inter-disciplinary community. Each year we have an All-Hands Meeting (AHM) in September. In 2004 we had about 680 people there. A third were the expected computer scientists. But the rest were drawn from a wide range of backgrounds — engineering, medical, computer services, industrial as well as from abroad.

Looking back, this involvement between the scientists and the computer practitioners has been a big advantage. If you look at Grid-like programs that exist, or have existed, at

Figure 2.6: myGrid



various times in other countries they were much more disparate. The applications, for example, were often being funded as one area while infrastructure was separately funded. This is changing, not least because others have seen what we have been able to achieve by having both applications and infrastructure work together.

Beyond 2006

Looking to the future, we have tried to put in place sustainable activities so as to ensure that, as the formal e-Science Programme comes to an end, activities and knowledge do not disappear and that services are still available for researchers. What we anticipate is that the Core Programme will continue for some number of years — but essentially at a maintenance level.

In addition, we have worked closely with JISC to ensure that those services that are required for the support of research are being picked up appropriately within the JISC programme. It is also possible that the Research Councils will offer more funding to various pilot projects, perhaps in particular areas or for continuing blue sky research. This is already happening. For example, the Engineering and Physical Sciences Research Council (EPSRC) has just put out a call for research on security as related to e-Science applications.

But the most important consequence is moving the exploitation of Grid technologies out into the wider community. I foresee these technologies and the e-Infrastructure they provide as just becoming a normal way of doing 'things' — whether in research or within a commercial context. Grid technologies will be integrated into the normal mechanisms for science research as we move forwards.

Lessons learned

One lesson learned is that if you can bring scientists, computer scientists and social scientists to work together there are many gains. It has been difficult at times to do this. At the beginning of the e-Science Core Programme I remember giving a talk about what we intended to a group of computer scientists and whose reaction was: "we are not going to be the programmers for the physicists". Similarly, when talking to the physicists their reaction was that "we do not need computer scientists, we can do our own computing".

Yes, there have been some difficulties in building an initial community but, having done that, we do see major advantages. One of the examples I can offer is a project called myGrid — where computer scientists and bioinfomaticians have been working together. They have created a set of specific services that support the bioinfomaticians in what they are trying to do. They no longer have to cut and paste from different Web sites; they no longer have to build little scripts to do jobs. They now have an infrastructure that supports them. And, because of this, they have new results coming out.

In talking to the computer scientists behind MyGrid they said "well when we went into this we thought we could just develop an infrastructure and then automate everything — the bioinfomaticians would just press a button and obtain results. But what we have learned is that the bioinfomaticians do not want everything to happen automatically. They need to have the tools to assist them in building their applications rather than automating the activity."

myGrid (Figure 2.6) is just one example of the transfer of understanding about requirements and skill sets that takes place. This has been a valuable lesson. This has led to a re-evaluation of the appropriateness of some CS technologies, to further research and in some cases to whole new CS research fields.

Building the UK e-Science Grid has taught us a lot — it isn't trivial! We have a much better understanding at this point regarding the middleware and service requirements and what it takes to run a production NGS service. We have also discovered that the typical UK project needs a 'plug and play composable services' Grid, which is rather different from the particle physics grid that is being developed globally.

An unexpected plus occurred at the time the e-Science programme started in 2001. At that time there was a significant downturn in the high tech sector and we anticipated difficulties in obtaining external support because industry would not want to invest in a research programme at a time when they were strapped for resources. What actually happened was that we saw more investment — not least because the certain industries looked at investments in Grid technologies as being a possible long term way of reducing their own resource requirements and decreasing time to market. Today, in the Inter-Enterprise Computing initiatives, we see industry benefiting from the pull-through of those investments, with the resulting efficiencies for which they were looking. In addition to the benefits from the advances that Grid technologies provide for science, and e-Science, the knowledge acquired does have relevance elsewhere.

One issue that has become very clear is the importance of standards and getting communities to agree common stan-

dards for interoperability — this model of shared resources is a difficult one to realize without such standards.

Now we need to put in place appropriate training and education for the new generations of e-scientists — because what we have learned is that they need different sets of skills than was traditionally true.

Finally, the most difficult problems that we have faced have not been the technical ones, but are the social changes required to allow this model of data and resource sharing.

Management conclusion

In this discussion Dr. Trefethen has described the e-Science Core Programme and what it has been achieving since 2001. As she describes, the achievements are many and varied and spread over diverse areas.

From the middleware angle, the e-Science Programme is still creating value, as input into the standards community as well as delivering open source middleware —which is available for both the scientific and commercial communities. Not only is the concept of Grids making progress, but it is coming closer to becoming a reality that we will all be able to use — if we can formulate the type of question which needs Grid-like resources to deliver a solution.

Web Services and the Grid

Tom Welsh
Consultant

Management introduction

Grid computing is one of the most exciting new trends in IT, promising to handle very large compute-intensive tasks and squeeze additional RoI out of existing desktop and server hardware. Before high end Grid computing can take off, however, the IT industry must agree on a suitable set of standards.

That is the mission of the Globus Project, an offspring of work done at the USA's Argonne National Laboratory and now rebranded as the Globus Alliance. Nearly ten years ago it created the open source Globus Toolkit, a suite of software that handles security, data and resource management and many other facilities for creating and running a high-powered Grid.

The next step looks like being the convergence of Grid computing with Web Services, as exemplified by Open Grid Services Architecture (OGSA). Worked out by IBM in co-operation with Argonne, the University of Chicago and the University of Southern California, OGSA envisages standards-based 'Grid Services' — with interfaces for:

- ***discovery***
- ***dynamic service creation***
- ***lifetime management***
- ***notification***
- ***manageability.***

It is being developed by the Global Grid Forum, another organization hosted by Argonne. Open Grid Services Infrastructure (OGSI), the core of OGSA and the part most closely related to Web Services, was published in June 2003.

Before OGSA and OGSI had a chance to stabilize, however, they were overtaken by a new wave of change. IBM, Globus and their various partners published no fewer than six related specifications in 2004, under the headings of Web Services Resource Framework (WSRF) and Web Services Notification (WSN). Since these explicitly supersede OGSI, Tom Welsh assesses where these events leave the Grid standards initiative.

OGSA and OGSI

From its foundation in 1995 until 2001, the Globus Project was dedicated to revolutionizing the application of massive computing resources to the solution of scientific, engineering and mathematical problems. This changed forever in August 2001, when IBM announced substantial investments in Globus' work. IBM and Microsoft joined the Globus Project in February 2002, at which time IBM and Globus jointly announced 'a set of new specifications that for the first time would allow businesses to share both applications and computing resources over the Internet'.

These specifications, which grew into OGSA, would help to 'move Grid computing beyond scientific and technical applications to real business applications'. All future releases of the Globus Toolkit were to be based on OGSA.

Simultaneously, IBM declared that OGSA would be a 'key foundation' to its autonomic computing strategy. Furthermore it promised a reference implementation — based on its own WebSphere J2EE application server.

What is OGSA?

OGSA aims to harness the power inherent in a Service Oriented Architecture to make it quicker, easier and cheaper to create efficient, reliable Grid applications. Its main objectives are to:

- **build large-scale systems in standard fashion**
- **facilitate the methodical re-use of code**
- **achieve interoperability between Grid components.**

The authors of OGSA warn that it is not meant to be a complete, ready to use, Grid environment. Like the Globus Toolkit — which has now become an implementation of OGSA — it (OGSA) represents just a set of tools which takes a lot of the hard work out of designing such an environment. But it still leaves designers great freedom of choice, and discretion.

Even with this proviso, OGSA is a massive undertaking. Its complexity stands as a warning to anyone who thinks that building enterprise-level Web Services is something that can be done quickly, easily or without heavy investment.

In a Grid system based on OGSA, there are at least four successive layers of abstraction:

- **Web Services (complying with specifications such as SOAP, WSDL, Web Services Security and the appropriate WS-I Profiles)**
- **the additional requirements of OGSI, which transform simple Web Services into 'Grid Services' (or, since 2004, of WSRF — which transforms Web Services into WS-Resources)**
- **the operations performed by whatever set of Grid services are required by the system**
- **custom logic to make each system perform the tasks for which it is intended, whether this be financial analysis or protein folding or whatever.**

The OGSA Architecture

A glance at Figure 3.1 shows that OGSA is a complex, heavyweight software system. Let us take a walk through the diagram.

The bottom two layers are called 'Physical and Logical Resources'. The lowest layer comprises physical resources — such as servers, storage and networks — all the physical equipment needed to run a Grid. These physical resources cannot be chosen at random; they are constrained by the requirements of OGSA. Eventually, no doubt, the J2EE requirement will be dispensed with, as OGSA is extended to other platforms (such as .NET).

At the second-lowest layer, the logical resources, the potential becomes more interesting. The logical resources are implemented in software, but in 'raw' form — they have no built-in Web Service or Grid interfaces. Thus, these can be used to construct Grid resources. But their work is done behind the scenes.

Among the necessary logical resources are big chunks of re-usable software with standardized interfaces — although these are more likely to be de-facto than de-jure. They include:

- **database management systems**
- **directories**
- **file systems**
- **messaging**
- **security**
- **work flows**
- **other building blocks.**

The next layer up (or the next two, depending on how you read the diagram) comprises Web Services and OGSI. The logical resources already mentioned are interconnected by Web Services through OGSI — which serves as a "Web Services to Grid adapter", supplementing Web Services with all the extra features required to implement Grid applications.

This takes us to the top two layers:

- **OGSA Architected Services**
- **Applications.**

Generally speaking, the architected services are those that Grid applications (the top layer) will call directly. The Global Grid Forum (GGF) has already done much work on defining architected services, and more are in the pipeline.

The Grid architected services fall into four reasonably homogenous main groups:

- **Grid Core Services**
- **Grid Program Execution Services**
- **Grid Data Services**
- **Domain Specific Services.**

Grid Core Services

As usual in these layered architectures, there are a set of common, relatively lower-level, services that support the more specialized services. The Grid Core Services comprise:

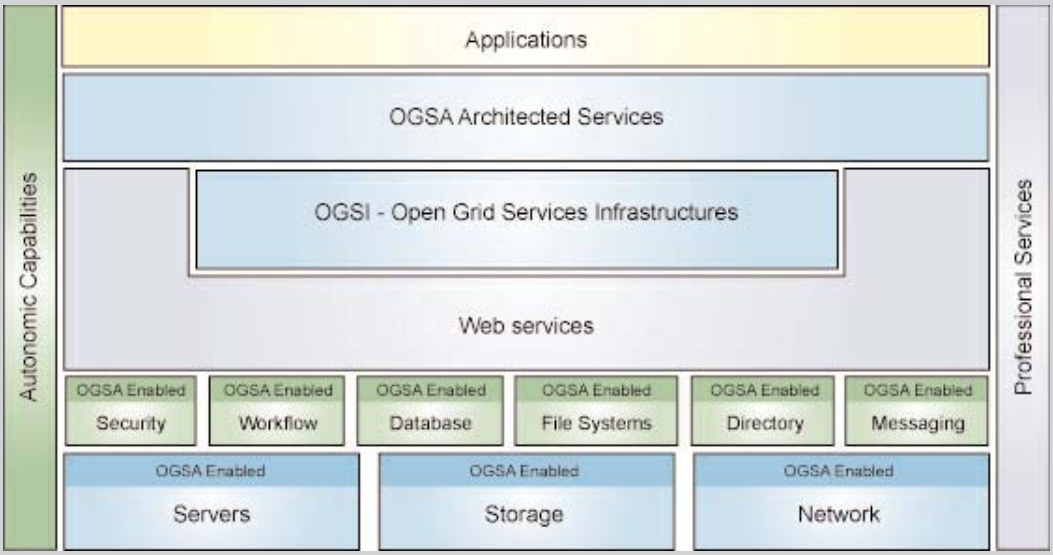
- **service management**
- **service communication**
- **policy**
- **security.**

These are all implemented as Grid Services, in the formal sense that they in turn build on top of the OGSI extended Web Services. In practical terms they provide the functions that Grid applications call on to deploy, initialize, start, stop and monitor programs and components within a distributed system.

Grid Core Services also handle:

- **sending out data**
- **receiving results**
- **storage**

Figure 3.1: OGSA main architecture (Source: IBM)



- **inter-service communication**
- **message queuing**
- **publish-subscribe event notification**
- **distributed logging**
- **and dozens of other essential tasks.**

Last but not least, they establish a framework for creation, administration, and management of policies and service level agreements (SLAs). These exploit the underlying security protocols and other mechanisms to provide authentication, authorization, trust policy enforcement, credential transformation and other security functions.

Grid Program Execution Services

Popular taxonomies of common Grid types often seem to see 'computational' Grids and 'data' Grids as fundamentally different architectures. As far as OGSA is concerned, however, computation and data management are equally important, and accordingly receive equal treatment.

The program execution services undertake tasks like scheduling and queuing jobs, reserving resources for specific execution threads, load-balancing, and micro-scheduling. Clearly, the business of scheduling is liable to become many times more complicated in a Grid, and sophisticated scheduler packages have been written to meet this challenge.

All in all, the Grid program execution services aim to 'virtualize' the loading, execution and termination of programs. In other words, the intention is to make it look as if applications are running on a single computer instead of a Grid.

Grid Data Services

To work effectively, a Grid needs to virtualize data as well as program execution. Application programmers should not have to go searching around the network, trying to find the information they need. Often, optimal or near-optimal resource utilization and efficiency can only be achieved by making decisions about storage, retrieval and routing at run-time. This means that the operating environment can be allowed to take care of these issues.

OGSA's Grid Data Services incorporate mechanisms for transparent storage and retrieval of data in distributed environments, using techniques such as replication, cacheing, and optimized data transfer. Furthermore, data may take many forms — for instance streams, files, documents and databases of various kinds — and may need to be transformed automatically from one type or format to another when necessary.

Domain-Specific Services

Little thus far has been documented about these services. Externally it appears that these have been left to be written by domain experts in various vertical industries, if and when they adopt OGSA.

OGSI

When the decision was taken to base OGSA on Web Services as a standard messaging infrastructure, it was felt that the existing Web Services standards had certain critical deficiencies for the given purpose. Among these were, for instance:

- **basic service semantics**
- **how services are created**
- **how long services continue to exist**
- **fault handling**
- **the preservation and management of state.**

These deficiencies were addressed through a set of standard interfaces called OGSI. An OGSI-compliant Web Service qualifies to be formally described as a 'Grid Service'. The latest version of OGSI is the 'Proposed Recommendation' published by GGF in June 2003 and Figure 3.2 shows a high-level overview of some of the main concepts within OGSI.

Inevitably, WSDL was used to define the interfaces and other characteristics of participating Web Services. In anticipation of WSDL 1.2, which had not yet been completed, some small extensions were made — resulting in Grid WSDL (GWSDL).

OGSI also provided behaviors and interfaces to control the full lifecycle of Grid Services, for instance:

- **service-level management**
- **collection management**
- **state change notifications**
- **service creation mechanisms**
- **instance-reference management.**

WSRF and WSN

Since it was envisaged right from the outset that OGSA would be based on Web Services, few should be surprised by the appearance of Web Service specifications specifically designed to support Grid computing.

The first such specifications — Web Service Notification (WSN) and the Web Service Resource Framework (WSRF) — were not published until January 2004. Akamai, The

Globus Alliance, HP, IBM, SAP, Sonic Software and TIBCO announced the new documents at the GlobusWorld conference in San Francisco. Part of IBM's press release read as follows:

"The WS-Notification specification and the WS-Resource Framework will provide a scalable (publish/subscribe) messaging model and the ability to model stateful resources using Web services. Stateful resources are elements that can be modeled including physical entities (such as servers) to logical constructs (such as business agreements and contracts). Access to these stateful resources enables customers to realize business efficiencies including just in time procurement with multiple suppliers, systems outage detection and recovery and Grid-based load balancing."

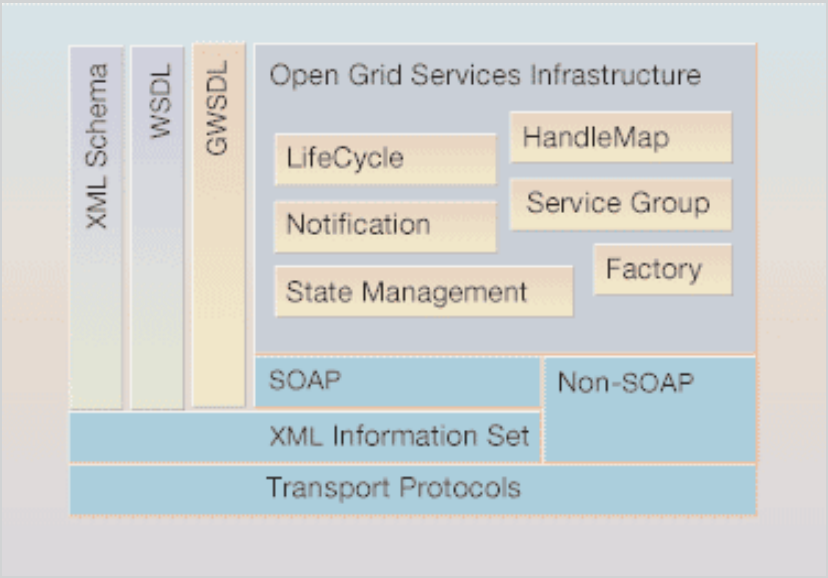
Justification of WSRF and WSN

Ian Foster, head of the Distributed Systems Lab at Argonne, and co-author (with Carl Kesselman) of 'The Grid' and 'The Grid 2', is one of the world's leading authorities on the Grid. When asked to explain the reasons behind the shift from OGSI to WSRF, he highlights four 'major concerns':

- **'Too much stuff in one specification' There was a feeling that OGSI was too big and inflexible and so the WSRF designers responded to this concern by creating a family of composable specifications, in the style of the IBM-Microsoft Web Services Framework; indeed, the WSRF specifications are composable with those of the Web Services Framework (such as WS-Security, WS-Coordination, and WS-Reliable Messaging)**
- **'Does not work well with existing Web Services tooling' One of the many obstacles to using popular Web Service development tools to build OGSA applications was felt to be OGSI's 'excessive' use of XML Schemas, which caused problems with standards like JAX-RPC (this was accordingly 'toned down' in WSRF)**
- **'Too object-oriented' OGSI encapsulated resource state within a Web Service, whereas WSRF draws a clear distinction between a Web Service and the stateful resources on which it acts**
- **'Introduction of forthcoming WSDL 2.0 capability as extensions to WSDL 1.1' GWSDL was an uneasy compromise between WSDL 1.1 and WSDL 2.0; on general architectural grounds, it was felt best to dispense with it.**

This is how Foster sums up the difference between WSRF and OGSI: "the changes from OGSI to WSRF are primarily syntactic but also represent some useful progress. The

Figure 3.2: OGSI components (Source: IBM)



separation of OGSI functionality into six independent specifications simplifies adoption, the use of WS-Addressing is a step forward, and less aggressive use of XML Schema and WSDL 2.0 features will facilitate the use of available tooling. Do these benefits justify the disruption associated with a revision to the Grid infrastructure specification? At a purely technical level, it's debatable, but the fact that we now have strong support in the Web Services community for Grid infrastructure is a huge achievement: it means we'll see WSRF support in core Web services products, which is wonderful news for the Grid community."

Reading between the lines of Foster's final sentence, it seems he was unhappy about the changes 'at a purely technical level' ... but acknowledges that there was no way forward without IBM's support.

Standardization

No sooner had WSRF and WSN been announced than GGF and OASIS hastened to publicize their co-operation. Their joint statement was short and businesslike:

"GGF and OASIS work together to review OGSI.

"OASIS develops Web Services standards that enable dynamic distributed systems. The GGF develops standards for Grid Computing, including the Open Grid Services Infrastructure (OGSI). A set of draft specifications revising OGSI has been created that leverages technology from both the GGF and OASIS. OASIS and the GGF have agreed to work together on open review and revisions of the specifications as part of an effort to connect more closely the Grid and Web Services communities.

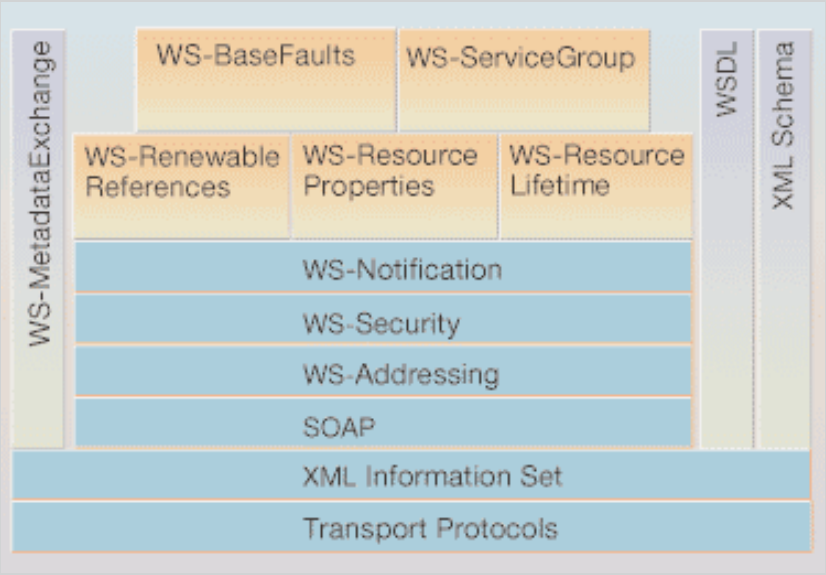
"Approved by the Grid Forum Steering Group and the OASIS Communication Staff."

The key fact was that OASIS and GGF wanted to work together, rather than competing or ignoring each other's efforts. This spirit of co-operation firmed up in March 2004, with the establishment of two new OASIS technical committees: the WSRF TC and the WSN TC.

Each of the new TCs was chartered to define 'a set of royalty-free, related, interoperable and modular specifications"'. In the year since their formation, neither TC has published any important new documents, or reported reaching any major milestones. Work continues, it must be assumed, on improving the draft specifications to the point where they can be proposed for adoption as OASIS Open Standards.

Figure 3.3 shows how (in IBM's view) the five WSRF specifications fit with earlier parts of the Web Services Framework. WS-Notification was initially treated as part of WSRF,

Figure 3.3: WSRF and other WS-* specifications on which it depends (Source: IBM)



but it soon cut loose (possibly thanks to the formation of an OASIS TC dedicated to it).

Implications for OGSA and OGSI

OGSA went back into the melting pot after the publication of WSRF. Fortunately, its previous architecture had not yet coalesced into anything as precise as a specification.

With the appearance of WSRF and WSN, the Web services foundation on which OGSA had been based began to shift under its feet. It had been overtaken by events — an outcome that we have seen before in the kaleidoscopic world of Web Services — and GGF took the opportunity to adjust its overall design at the same time. Hence, the current version of OGSA amounts to little more than a set of use cases and a preliminary draft specification, which bears equally little resemblance to the architecture diagrams (for example, Figure 3.1) floated in 2003.

There is also an OGSA Glossary, whose contents yield some useful hints about the project's shifting premises. The definition of "Grid Service", for example, is as follows:

"1. (deprecated) In OGSI, a Grid service is a service that implements the GridService portType. This use of the term is considered to be deprecated.

2. (informal) In its more general use, a Grid service is a Web service that is designed to operate in a Grid environment, and meets the requirements of the Grid(s) in which it participates."

Even more telling is the entry for OGSI:

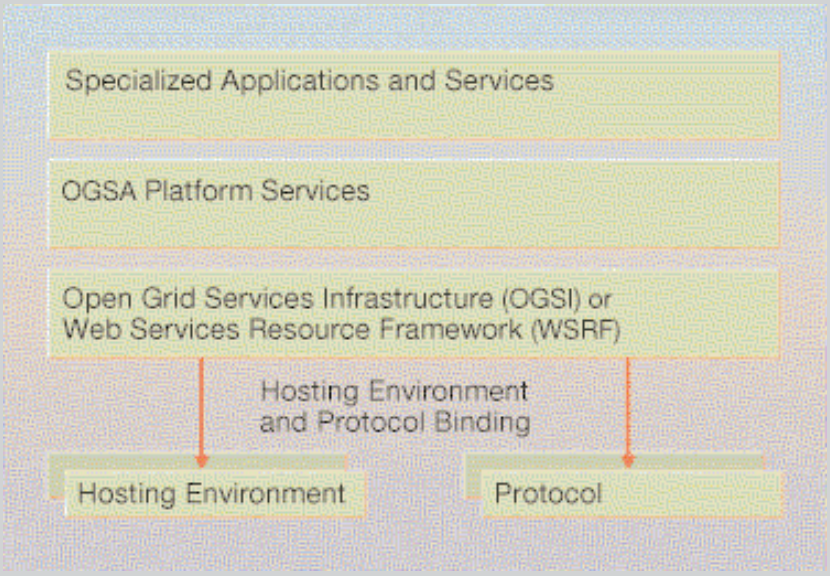
"Open Grid Services Infrastructure. A GGF specification that defines the common interfaces and behaviors of a Grid service. OGSI is deprecated in favor of WSRF and WSN."

WSRF and WSN have not imposed any major structural changes on OGSA. They merely replace OGSI, doing the same things in slightly different ways; all that is required is a shift in syntax.

This point of view is illustrated by Figure 3.4, which looks rather like a shrunken version of Figure 3.1. The top two layers — Applications and Architected Services — are more or less unchanged, except that the Architected Services are now called OGSA Platform Services (a name that describes their function rather more precisely).

Comparing the middle layers of the two diagrams, however, we find that the 'Web Services' and 'OGSI' layer is now labeled 'OGSI or WSRF'. This reflects the belief that the two are 'plug-compatible', and that it is feasible to remove all references to OGSI and substitute references to WSRF and WSN.

Figure 3.4: The new OGSA, using either OGSI or WSRF (Source: IBM)



As Foster explained in an interview with IEEE Distributed Systems Online: "the WSRF proposal is a refactoring of OGSI concepts to align better with Web Services. The need for this refactoring became apparent when Web Services vendors indicated that while they recognized the importance of OGSI concepts, they would not adopt OGSI as it was then defined. This response threatened to prevent the ubiquitous support for Grid infrastructure that was a primary motivator for OGSI. So, while we in the Globus Alliance had little enthusiasm for revisiting the OGSI design (we'd much rather be developing higher-level services than fiddling with infrastructure), we concluded that some refactoring was justified if it could affect the convergence of Grid and Web Services. We teamed up with Web Services architects to study this issue, and the result is WSRF."

Management conclusion

Once the decision was taken to base OGSA on Web Services, the OGSI layer became necessary — to mediate between:

- ***raw Web Services***
- ***the more sophisticated Grid services needed to power OGSA.***

The subsequent changeover to WSRF and WSN did not change this decision, in principle, but it did force the OGSA architects to go back to the drawing board. Less obviously, but with potentially damaging effect, it made OGSA dependent on a whole sheaf of Web Services specifications — which had not been the case before.

It may take a few years before those Web Services specifications become stable enough for big, complex systems like OGSA Grids to be based on them. Until then, look forward to occasional 'reshuffles' that will destabilize any systems built on top of Web Services. That old principle — that warns against adopting immature standards — still holds good.

Although the authors and supporters of OGSA insist that it is not meant to be a ready-to-use framework for building Grid applications, few can deny that OGSA is a most impressive piece of work. Equipped with its extensive array of features, future Grid developers should find the challenges facing them a great deal easier to tackle.

OGSA also constitutes an excellent testbed for Web Service concepts. If huge scientific and commercial Grids can be successfully deployed and administered using a software infrastructure composed of Web Services, many questions about their viability will be laid to rest.

Managing and monitoring Web Service performance

Mark Lillycrop
Principal Analyst
Arcati

Management introduction

In today's business world, organizations are becoming more and more reliant on Web Services in all their various forms — from web-based applications and applets and the middleware that supports them, from Web front ends to back end office systems through to transactional systems with some Internet-based component (often poorly defined). What these services frequently have in common is a rather informal approach to operational management, certainly informal in comparison to the internal corporate systems with which they interact.

Managing these disparate new technologies, and ensuring that Web-oriented resources deliver a level of service that can be monitored, measured and geared to business requirements, remains one of the biggest challenges for IT departments today. In this analysis, Mark Lillycrop examines and assesses what has been happening, and what has not, in this key space.

Traditional data centers

For the traditional data center of a few years ago, service management was a reasonably predictable process:

- **capacity planning was a well understood discipline**
- **applications were allocated system resources according to pre-defined priority levels**
- **help desks were sized to cope with a predictable number of customer queries**

Barring unforeseen disasters, service delivery was relatively consistent.

A different environment

Today, the managing of services end to end is a different proposition altogether. While many talk about Service Oriented Architectures (SOA), the 'S' all too often refers to the speed with which new services and applications can be created on the fly rather than the extent to which the service can be measured or predicted.

With Web Services, performance degradation can have 101 different causes, from unexpectedly popular marketing campaigns to denial of service attacks or unassociated bottlenecks 'somewhere' on the network. Parts of a given Web Service might fall within an enterprise's own established management framework but other — equally critical — components may be invisible to traditional enterprise management tools. In such cases, the user — and even the administrator — of the Web Service may have no idea why the level of performance is declining. This can have a serious knock-on effect on business efficiency.

A certain amount can, of course, be done to avoid performance degradation — for example, over-provisioning of bandwidth, processing power and storage can mitigate the effects of heavy traffic on Web performance. Most IT vendors now provide capacity-on-demand services in order to help customers deploy extra resources as and when required.

But increasing capacity is only part of the answer. Some applications can peak at 50 times their normal traffic levels (think of the SuperBowl and other similar events). Today's businesses have to accept that it simply is not feasible to provide acceptable performance to all Web Services, all of the time.

Instead, we need to find more sophisticated ways of managing the way that Web Services perform. This must focus particularly on the inter-enterprise middleware, for it is this that integrates the various parts of an application (Figure 4.1).

The building blocks of Web Services

The main problem with Web Service performance is that component technologies are not ideally suited to a business critical workload. HTTP, the core protocol of all contemporary Web activity, is a best-effort delivery mechanism. If there is any limitation on bandwidth, HTTP will simply discard packets.

Any effort to improve on its performance by using clever application design depends on higher level protocols (such as HTTPR, BEEP and DIME). But these have been developed and adopted much more slowly in the business world.

Asynchronous messaging mechanisms — such as JMS or MQSeries — provide a much more reliable approach to designing Web Services. That said, even these introduce management issues, in terms of overall response times.

Similarly, XML and the Simple Object Access Protocol (SOAP), which have gained widespread popularity because of their business focus and near universal support, are extremely resource-intensive. They bring with them a huge processor overhead and an irrepressible thirst for bandwidth. These demands are placing far greater pressure on network resources than more traditional middleware components have done.

There are a number of techniques — that are currently emerging — for software designers to make XML-based applications more resource-efficient. But these are not yet ready for real use. In the meantime, performance and reliability need to be carefully managed and monitored.

Dedicated or plug-in management?

One of the key decisions that any large enterprise needs to make as it moves into Web Services is whether it (the enterprise) needs to manage these services discretely or whether these (services) should be controlled through existing enterprise management consoles.

According to Tom Murphy of Web Service integration specialist Cape Clear Software, there are two main schools of thought. "You may take the view that Web Services management requires dedicated software systems and consoles that look after all aspects of managing, billing, and securing your services. This is seen by many as a standalone opportunity for new Web Services management players — such as AmberPoint, Actional, etc."

He goes on: "On the other hand many organizations do not want yet another management console. Instead, over time, Web Services management will be handled as part of the existing management infrastructure — using CA's Unicenter (Figure 4.2), IBM's Tivoli etc.

"From working with over 200 customers, we believe that the latter is the most likely outcome. For example, as part of our own ESB, we provide customers with a Web-based management console where they can track, start, stop, secure, etc. their deployed services. However we also ship SNMP and JMX (Java Management Extensions) adapters in the box, which means customers can link this services management piece into an existing management infrastructure."

For many IT users, deciding which way to go is a difficult decision to make right now. Web Services are still in their infancy, compared to the many long-life aspects of a corporate IT infrastructure. The jury is still out on the most effective ways to improve the manageability, security and performance of Web Services. Depending on how responsive an organization needs to be to variations in Web Service performance, it may even choose to select a more specialized approach to tooling.

Cape Clear's John Maughan puts it like this: "The problem of Service Level Agreement management can be split in two:

- **monitoring the interface and detecting problems**
- **doing something about the problems that you detect.**

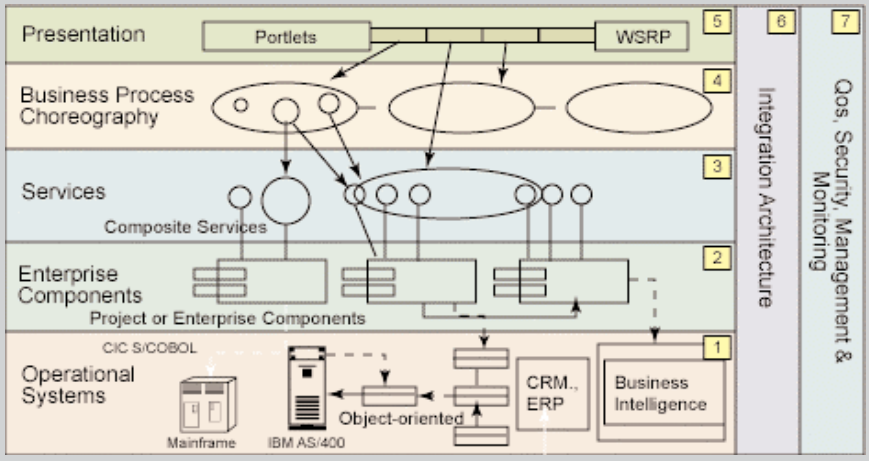
"The former problem is a pretty easy one to solve. It is possible with most management consoles to monitor performance and to set performance and SLA thresholds for alarms etc.

"The latter problem is where there are gaps in existing solutions. First of all — when you know there is a problem — you need to be able to drill down into that problem in order to find the root cause. This is the area in which Web Services Management 'pure-plays' have a slight edge over traditional enterprise system management solutions.

"Secondly you need to be able to provide automatic dynamic failover, load balancing and re-prioritization capabilities, and there are few solutions which (yet) provide dynamic facilities like these. This is most likely to be because organizations build in the appropriate levels of redundancy capacity to support their SLAs. These levels are reviewed periodically (but not dynamically)."

In other words, organizations that need to react rapidly to specific performance problems within the Web Services environment may benefit from the emerging generation of tools from the dedicated Web Services management ven-

Figure 4.1: Managing the way Web Services perform



dors. The traditional monitoring tools — from the likes of IBM/Tivoli and CA/Unicenter can identify bottlenecks, security exposures and network faults via SNMP or JMX interfaces. But they offer little scope to drill down further and identify the precise cause of a problem (and potentially the solution).

However, as Maughan points out, even specialty Web Services products provide little scope for automating the management process and responding to performance problems by adding capacity on the fly.

At the current level of maturity for these products, users need to accommodate the capacity requirements and availability characteristics of Web Services through careful planning by building sufficient flexibility into any relevant service level agreements.

The key question

For suppliers such as IBM/Tivoli and CA/Unicenter, the questions are really:

- **how much control the user needs over his or her Web Services**
- **what level of information is needed to make forward planning decisions.**

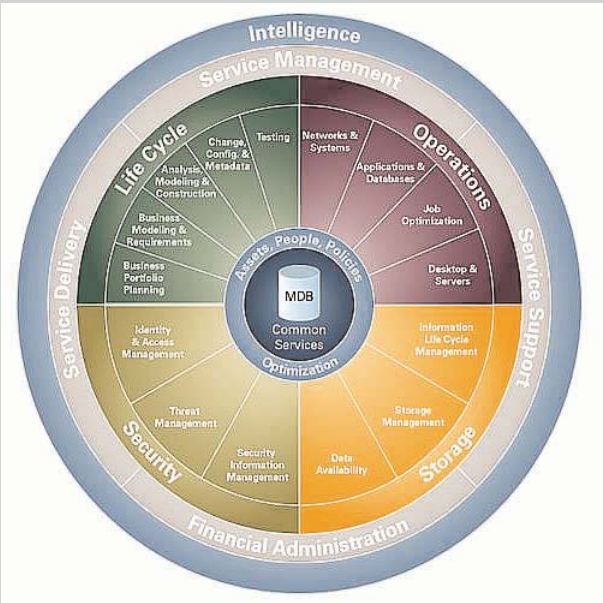
IBM/Tivoli argues that services need to be viewed within the context of the broader IT infrastructure (Figure 4.1). In this scenario, there is a mature operational management layer at the bottom, with a clear picture of the systems and databases at the heart of the core enterprise business systems. Containers, transactions and application components are monitored and managed at the next level up. At the top level are the presentation tools — and the increasingly sophisticated business work flow products that align IT resources with physical business systems.

Sandwiched between these two tranches of management technologies are the services. "In many ways, services are an abstraction," argues Tivoli's John Knutson. "How you manage that abstraction will depend to a large extent on how those services relate to other IT components. You may need to drill down and up, zoom in to understand the precise performance characteristics of a service or zoom out to see the broader picture."

Knutson points out that, with Web Services, many management decisions are about routing. You may have policies in place to determine that, if one service or data source is consistently unreliable, another route will be taken. Some decisions of this kind are very complex, and the more information about IT and business events that can be fed into the decision-making process, the more efficient the Web Service environment will be.

Much of this process can be automated over time, as businesses move towards a mediation approach. In this, changes to message content, routes and even formats can

Figure 4.2: CA's Enterprise Infrastructure Management



be made on the fly — according to pre-defined service management goals. All this requires a fairly high degree of integration of Web Service management tools with other parts of the IT management infrastructure.

Right now, though, there is no simple way of doing this. The traditional management frameworks still rely on specialist solutions to provide the drill-down capability into Web Services, which means that there is a burgeoning opportunity for integrators to put together the right tools to help customers manage their Web Services effectively. In the not-too-distant future, though, much of this function will likely be absorbed within the broader system management solutions. Many enterprises are, therefore, currently adopting both a tactical and a strategic solution to Web Service management. At the current level of adoption and sophistication, they are taking a tactical approach to Web Service management by observing performance and reliability over time and making sufficient resources available to their systems to compensate for any inconsistency in service levels.

In the longer term, however, as these applications become more critical to the business and more information is needed on interdependencies between services, users will move towards tighter integration. In all probability this will either be through an individually tailored point solution or by expanding the functionality of pre-existing frameworks.

Planning for Web Service performance

The essential element in Web Service management is knowing what to expect. Without the reassurance of traditional management techniques — which monitor each system, network or database event and send alerts when potential problems loom — everything comes down to planning and preparation. This means understanding how each Web Service:

- **interacts with other IT resources**
- **is likely to behave (based on past experience)**
- **can have realistic service levels set, without the benefit of mature service measurement tools.**

Writing in a recent issue of Web Services Journal, Fred Carter, Chief Architect of AmberPoint, stresses the importance of Web Service management pilots as a way of gauging realistic service management expectations:

"It's important to incorporate some Service Level Management (SLM) into your Web Services systems from the outset. Pilots and proofs of concept, which help to form your overall Web Services strategy, call for careful evaluation and constant fine-tuning. With SLM, you and your team will benefit from a deeper understanding of system performance. What's more, you'll need SLM-generated data to quantify your successes in order to justify additional investments in Web Services."

This is an extremely important point. Many organizations are still feeling their way with Web-facing applications and with the middleware that integrates these various resources with other IT systems, whether inside or outside the business.

This is unknown territory. Only by experimenting via pilot systems — and gathering performance data over an extended period — can organizations build up a meaningful picture of the way that their use of Web Services is likely to behave under normal working conditions. Proofs of concept always have an important role to play, but in this case they are essential in order to ensure than sufficient system and network capacity are provided and that any potential bottlenecks and areas of vulnerability are identified well in advance.

Differentiated service levels

Providing a consistent quality of service across the board is an achievement in itself with Web Services. A far greater challenge, however, is differentiating between service levels.

The Web has become a victim of its own success as a way of conducting business. With 24x7 availability, the Web is subject to numerous peaks and troughs in traffic levels. It is notoriously difficult to differentiate between the service level experienced by high- and low-value customers.

Even with a simple Web transaction, this is a major problem. In many commercial businesses, it is not uncommon for the top 5% of customers to generate 95% of profits. Abandoned transactions, due to poor response rates, are estimated to be worth some $25 billion a year — and there are numerous projects underway to provide a guaranteed service level to the 'super-users' who are most valued by the business.

This is not just a commercial issue, of course. Government information sites, for example, need to be accessible to all. But in the event of a national emergency it might be necessary to provide a fast-track route for civil defence and emergency service managers at a time when traffic levels reach an all-time peak.

With complex Web Services, of course, the issue of service differentiation is an even greater challenge. Now your super-users are no longer people; they are other services. As more and more Web Services interact, it becomes ever more critical to ensure that tightly managed mission-critical applications are not left at the back of the queue when requesting data or processes from other systems.

This takes us back to the Web Service pilots. Before going live with any new system, enterprises must have:

- **have detailed performance statistics in place for every other service that it touches**
- **use tools and procedures available for pin-pointing potential problems**
- **make sure that any Service Level Agreements take into account not just the key applications that come within their own enterprise management architecture but also those services over which they have little or no control.**

Techniques for managing service levels

Web Services still have a long way to go before they can offer the differentiated 'class of service' approach that IT (and its customers) have enjoyed for many years when using traditional internal corporate systems. Part of the problem is that the TCP/IP networks that underpin Web Services were never designed with differentiated service levels in mind. Mainframe-based network standards (such as IBM's SNA), which were growing up at the same time in the commercial environment, had quality of service at their heart, and it was commonplace twenty years ago to build class of service criteria into networked business applications.

But the Internet has remained, resolutely, an egalitarian standard — with one level of service for all. While there have always been facilities within TCP/IP for routing different traffic types at different speeds or across different parts of the network, these remain poorly defined and are, for the most part, neglected by the business or enterprise community. Facilities such as RSVP and DiffServ — in later versions of IP — are intended to address this problem. But these focus much more on physical traffic types — like file transfer, Voice Over IP, etc — rather than on policy-based business distinctions. The Internet Engineering Task Force, with its Policy Framework and Common Open Policy Service, has also explored ways of putting more prioritization control in the hands of Web application designers, and providing real differentiation between service levels.

For the time being, however, there are a number of techniques available to designers who want to fine-tune the service levels offered by Web Services. For example, Fred Carter recommends using XML syntax to provide more detailed information on business priorities to your Service Level Management tools:

"Tags, attributes, and element structure, as well as data from external sources such as LDAP, provide valuable contextual information that can augment use of the message content. By leveraging the content (for example, order value) and context (for example, customer priority) of the messages that are exchanged between Web Services, the management solution can provide valuable business insight into service-based processes."

Once again, though, it becomes clear that the process of managing service levels within the Web environment is a constantly moving target. Web Services and the relationships between them are in a constant state of flux, and it is essential for system managers to:

- **examine and re-examine the performance data and other management statistics available to them**
- **use this information to keep Web Services appropriately aligned with changing business priorities.**

Management conclusion

In many ways, Web Service applications and middleware are the latest phase in a constant evolutionary process within enterprise IT to integrate new technologies and function into the existing management infrastructure. Like earlier waves of business-to-business e-commerce, Web Services have been standardized at the lowest level, and lack the proprietary richness of management capability that characterize many internal IT systems.

Where Web Services now go beyond basic e-commerce systems is that they are essential to the success of emerging on-demand IT services. These are widely expected to change the face of the IT landscape over the next ten years.

Because of this, expect to see substantial resources being invested in Web Service management in the months and years ahead. This is likely to be accompanied by the rapid maturing of many of today's tools and technologies, which look distinctly rudimentary when compared to traditional system management capabilities.

Thoughts on Grids

Peter Bye
Consultant
Unisys Systems & Technology

Management introduction

In this analysis Peter Bye takes the view that Grid computing does possess a number of attributes that are not present in the great majority of inter-organizational networks. However, he also argues that Grids are a natural extension of the kind of environment that is already being constructed from Web Services.

In essence he believes that Grids are not something totally different. For example, one can think of distributed computing environments spanning a broad spectrum of sophistication. At the simplest end of the spectrum, he suggests that we may have just interactive use of applications by workstations of various kinds. As one moves along the spectrum, one can include additional ways of interacting, moving through Web services and eventually into Grid computing.

Middleware provides the services and tools needed to construct and operate such distributed environments. Thus Grid technology is realized in middleware developments. This analysis, therefore:

- ***looks at the origins, attributes and architecture of Grids***
- ***considers some implementation models.***

Grids attract attention

There has been an upsurge in interest in Grid computing in the recent past. From its origins in the scientific and academic communities, the interest has spread into the commercial arena, addressing a variety of requirements and suggesting new models of organization. For example, the notion of a virtual organization formed from a number of autonomous enterprises has attracted attention. (For some reason much of the literature spells Grid with a capital G; this convention is followed here.)

This raises a number of interesting questions. We, in the IT industry, have been developing a variety of models for inter-enterprise co-operation for quite some time, as the plethora of <letter>2<letter> acronyms suggests:

- **B2B**
- **B2G**
- **G2G**
- **G2B**
- **and so on.**

Service-oriented architectures — implemented using Web Services technology and built around XML, SOAP and other technologies — greatly facilitates inter-enterprise collaboration across the Internet (or an extranet). This has enabled the creation of sophisticated networks across which services may be cascaded, where a requester invokes a service — which in turn may invoke other services that are unseen by the original requester.

The relatively loose coupling of Web Services means that individual services can be implemented in any way the implementer chooses, as long as these conform externally to the Web Services 'rules'. Implementers are not therefore confined to any specific operating system or middleware infrastructure within their own environments.

So, one logical question to ask is 'what is a Grid'? Other relevant questions include:

- **when would an inter-organizational network built around Web Services become a Grid?**
- **is this a Grid already?**
- **does this depend on the number of systems involved: if more than n systems constitutes a Grid, what is the value of n: 20, 40 or?**
- **can any distributed environment be thought of as a Grid?**

In other words, is the word 'Grid' now inherently meaningless? These questions and others have been asked and answered in various ways, as will be seen later.

Origins and attributes: the definition of a Grid

Grids first became of interest in the early 1990s in the context of scientific and engineering work. Ambitious computer projects were beyond the limits of technology then available, so the people involved became interested in finding new solutions.

Three classes of applications were of primary interest (Source: "An Ecosystem of Grid Components", the Grid Research Integration Deployment and Support Center [GRIDS]. Go to http://www.Grids-centre.org and select Grid Ecosystem which is a Web site that contains a wealth of information and references to other sources):

- **computation intensive applications, including interactive simulation (climate modeling), very large-scale simulation and analysis (galaxy formation) and engineering (linked component models); the ability to harness a large number of computers to work on a single calculation is one way of obtaining super-computer power**
- **data intensive applications, including experimental data analysis (high-energy physics) and image sensor analysis....**
- **distributed collaboration applications, including online instrumentation (microscopes, x-ray devices etc)....**

Important common characteristics of each class of these applications were that the projects:

- **were large-scale**
- **were ambitious**
- **required several organizations to collaborate, sharing computing and other resources.**

Those involved found that there were a number of equally common problems that appeared in each project. Many of these problems arose from the fact that a number of different organizations were involved, each possessing different security, administrative and other policies and procedures.

That said, however, it was easy to see how the specialized requirements of such scientific applications might be extended more generally into the commercial world. Indeed, there what has become clear over time is that there is no clear boundary between the two. For example, the development of large-scale products, such as commercial or military aircraft, is a commercial venture — but it is also very technical. It requires the collaboration of many different organizations, often spread over a number of coun-

tries; the European Airbus Consortium is one good example.

Organizations involved in ventures such as this form a virtual organization, for which Grid technology is an enabling tool. This notion was originally developed by Foster, Kesselman and Tuecke (*Ian Foster, Carl Kesselman and Steven Tuecke "The Anatomy of the Grid: Enabling Scalable Virtual Organisations", Intl J Supercomputer Applications, 2001*). They explained what they referred to as the 'Grid problem': it is "flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources". Providing solutions to these problems enables the virtual organization to start and to exist.

Foster returned again to the subject of the definition of the Grid (*Ian Foster 'What is the Grid? A Three Point Checklist', Argonne National Laboratory & University of Chicago, July 2002*). His thesis was that, because interest in Grids extended from the academic to the more popular worlds, the term risks becoming meaningless. (He cites a source as defining the Grid as a 'funding concept'.) He also refers to an earlier book (*Ian Foster & Carl Kesselman 'The Grid: Blueprint for a New Computing Infrastructure', 1998*) and to the paper cited earlier (*The Anatomy of the Grid*).

In the 2002 paper, Foster says that "a Grid is a system that:

- **co ordinates resources that are not subject to a centralized control...**
- **...using standard, open, general-purpose protocols and interfaces...**
- **...to deliver non-trivial qualities of service."**

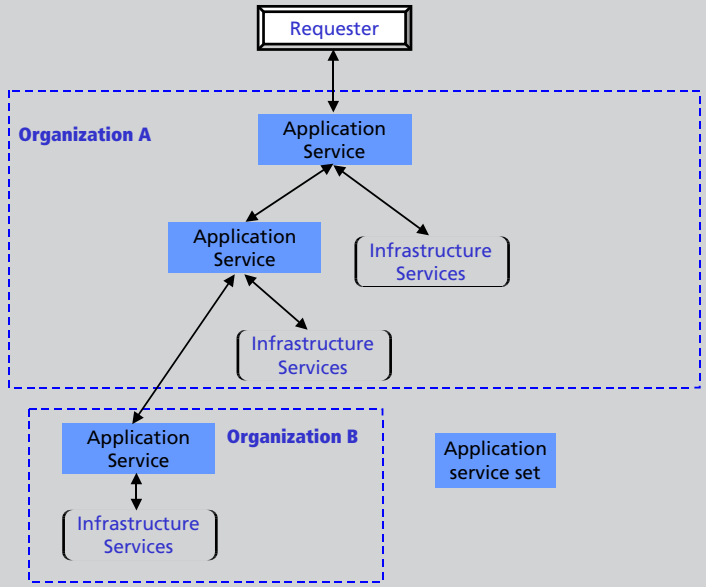
Foster (and others) cite examples of systems that are like Grids, and are sometimes referred to as such, but are not really Grids according to the above criteria. Some, for example, have fully centralized control of resources. He also says that the Web is not (yet) a Grid because it lacks the co-ordinated use of resources, although it does support access to distributed resources using standard protocols.

Resource co-ordination is a key point. As was noted earlier, collaboration among different enterprises is not new: it has been done at some level for decades, for example among airlines. Organizational electronic collaboration was also developed through EDI (Electronic Data Interchange). More recently, it (organizational electronic collaboration) has received massive amounts of attention through developments in Web Services technology. Thus the question remains: when does such collaboration become a Grid?

A service-based view

To throw some light on this question, consider different levels, or types, of collaboration using a simple example of

Figure 5.1: Service and service set relationships



two organizations, A and B. The analysis makes use of service concepts and employs the following definitions:

- **Service: in this context, a service is a software entity, which delivers something specific to a requester of the service; a service is defined by the protocol used to interact with it and its behavior — what it delivers — in response to various protocol exchanges**
- **Application Service: this is a service which is concerned with what a business does (getting the balance of a bank account and checking the status of a credit card are simple examples); because services can be combined in different ways, and can be distributed over a number of servers, the notion of an application is more complicated than it used to be and the term application service set has been used to identify one or more application services, which are related at a business level (a core banking system, for example); furthermore, application services can appear in more than one application service set**
- **Infrastructure Service: this is a service provided to support the environment in which the application services exist; it makes it possible for application services to be executed (examples of infrastructure include storage, networks, processor and security services).**

As can be seen from the above, application services depend on and make use of infrastructure services. Application services may use others, which in turn may use yet others. Figure 5.1 is a schematic of these relationships. It shows the two organizations, A and B.

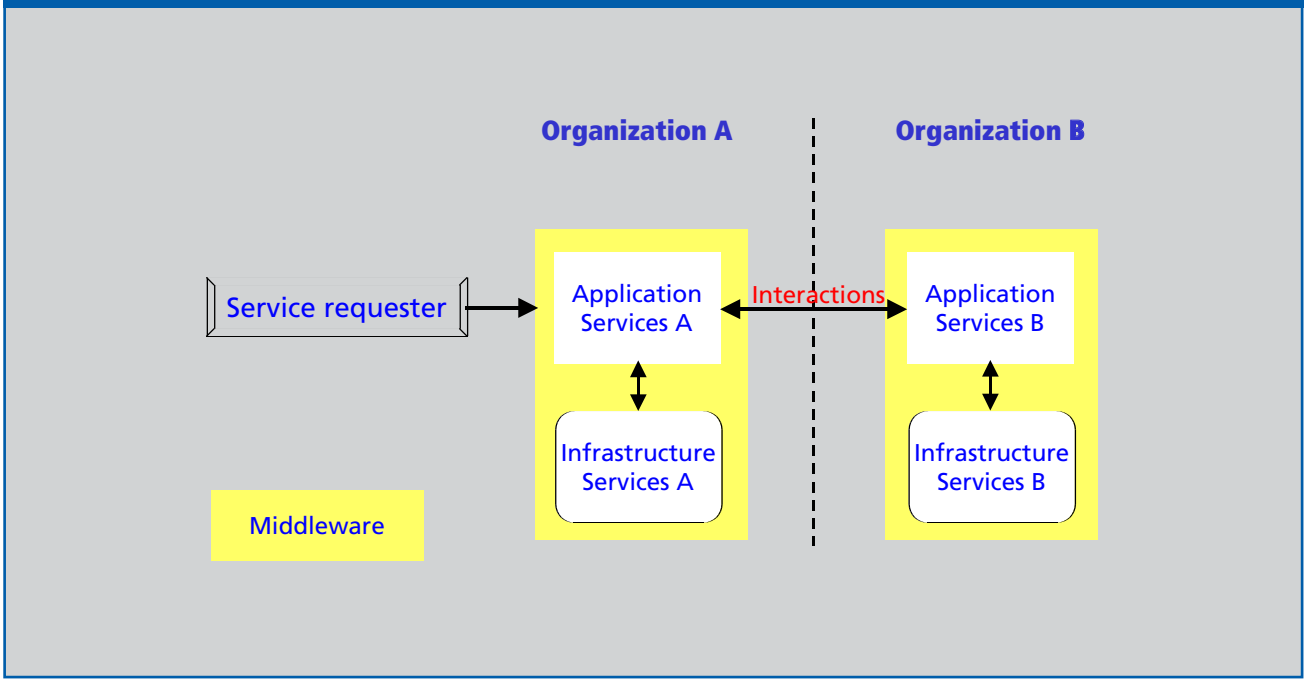
A requester invokes an application service in organization A. This then invokes another application service in the same organization, which in turn invokes an application service in organization B. All the application services use infrastructure services. The collection of application services may be thought of as an application service set.

Consider next a fairly simple interaction between services located in organizations A and B. Figure B.2 shows this. Organization A has application services. These use infrastructure services provided within organization A. Organization B also has application services, with supporting infrastructure services.

The requester invokes application services in A, which in turn invoke application services in B. The application services in each organization use their local infrastructure services.

A simple example is one where (say) A is selling things over the Internet. The requester is someone buying one of A's products. The requester provides credit card details, which are sent to B, a card issuer validating card details.

Figure 5.2: A simple interaction between application services in A and B



The picture can easily be extended to include services in organization A invoking services in other organization — such as suppliers and transportation companies. Logically, all inter-organization interactions might be made across the Internet using Web Services technology.

The interactions between application services, and between application and infrastructure services, are provided by middleware — and possibly other software not usually classed as middleware, such as operating systems and databases (the middleware infrastructure is shown by the shaded boxed in Figure 5.2). Middleware supports Web services (SOAP and so on) for the inter-organization connections.

Does the above system constitute a Grid? The answer has to be no because, although standard protocols are used and services are invoked across organizational boundaries using standard protocols, there is no co-ordination of other resources, especially infrastructure. Organization A would need to satisfy any security requirements imposed by B — for example, sending encrypted messages and using security certificates — but that is all. In Figure 5.2, the interactions between application services and infrastructure services are confined within each organization.

More complex examples

Consider now some more complicated cases. Suppose first that organizations A and B are collaborating on some complex venture. Organization A is going to send B a large quantity of information requiring a lot of processing power and network capacity.

Successful collaboration will depend on each of these organizations having sufficient infrastructure available — processor power and network bandwidth in this case — for the collaboration to be successful. If the resources cannot be reserved, the collaboration will not continue, as the results will not be useful. An obvious example of this might be reserving bandwidth with appropriate Quality of Service for a conference exchange involving video, voice and data. Without the required bandwidth, there is no point in continuing.

To manage the resources, application services in A may request local infrastructure and also remote infrastructure — using a middleware interaction (Figure 5.3). An alternative could be for the application services in A to request local infrastructure services, which then request remote infrastructure in organization B.

Depending on the results, the collaboration may go ahead, or the requester will be informed that the resources are not available and the proposed collaboration would then be postponed and rescheduled. This may or may not require a human decision. Some interactions could be postponed and automatically rescheduled when resources become available. What is possible depends on:

- **the purpose of the interaction**
- **the richness of the infrastructure services and supporting protocols.**

Consider now a different example. Suppose that organization B provides infrastructure services — processor and storage, for example. A client — organization A in this case — could choose to use this infrastructure to execute application services, either because its own infrastructure is overloaded, temporarily partially disabled or as a matter of business policy. In other words, such clients would farm out work to an infrastructure provider according to various internal circumstances or policies. In some senses, which we shall return to later, the infrastructure provider can be viewed as a utility company.

The work executed by organization B on A's behalf would be free-standing, in the sense that the original requester just wants the results. Or it might be part of a wider, distributed application service set, part of which is run in organization A and part in organization B.

How can this be achieved? One way would be to regard organization B as a kind of bureau (a long-established idea). Organization A would provide the application services and associated datasets to B, which would install them on suitable equipment. The two organizations would then agree on service levels and the criteria according to which A would invoke the execution of the application services in B's infrastructure. Various conditions could be imagined, for example business continuity or additional resource in the case of temporary overload.

The above approach, while useful and widely employed, does not conform to the notions of a Grid. In fact, it reduces to the example shown in Figure 5.1, with the possible additional conditions that the application services located in organization B may not always be executed there but could be executed in organization A under some circumstances.

The critical point is that the resources required are agreed in advance and not subject to dynamic allocation to any significant extent. It is of course possible that organization B could host a number of application service sets on behalf of different clients, but could not execute all of them simul-

taneously. For example, if it hosted business continuity systems for several clients, it is unlikely that all of them would require the service at the same time. But allocating the resources is hardly a Grid-like activity.

Yet more sophistication

A more sophisticated approach would be for organization A to regard B as a utility in the sense that electricity is a utility (this could lead to a dangerous confusion of grids). If organization A decided that it did not have sufficient infrastructure resources to execute application services of some kind, it could use the resources of B — perhaps shopping around if there are alternative organizations providing the same services.

The process would be something like the following. Requests for infrastructure in organization A would show that there is a need to get extra resources externally. Appropriate interactions would then take place with organization B to identify and reserve the resources. Having established a suitable source of infrastructure in B, the next step is to get the application services and any datasets required to B's environment.

This pre-supposes, of course, that B's infrastructure is logically able to run the applications. A simple example would be that A has services written in a specific programming language, which are compiled and executed in A's infrastructure. For B to provide a suitable infrastructure for A, B has to be able to compile the services or have the same hardware and environmental platforms to execute the object code.

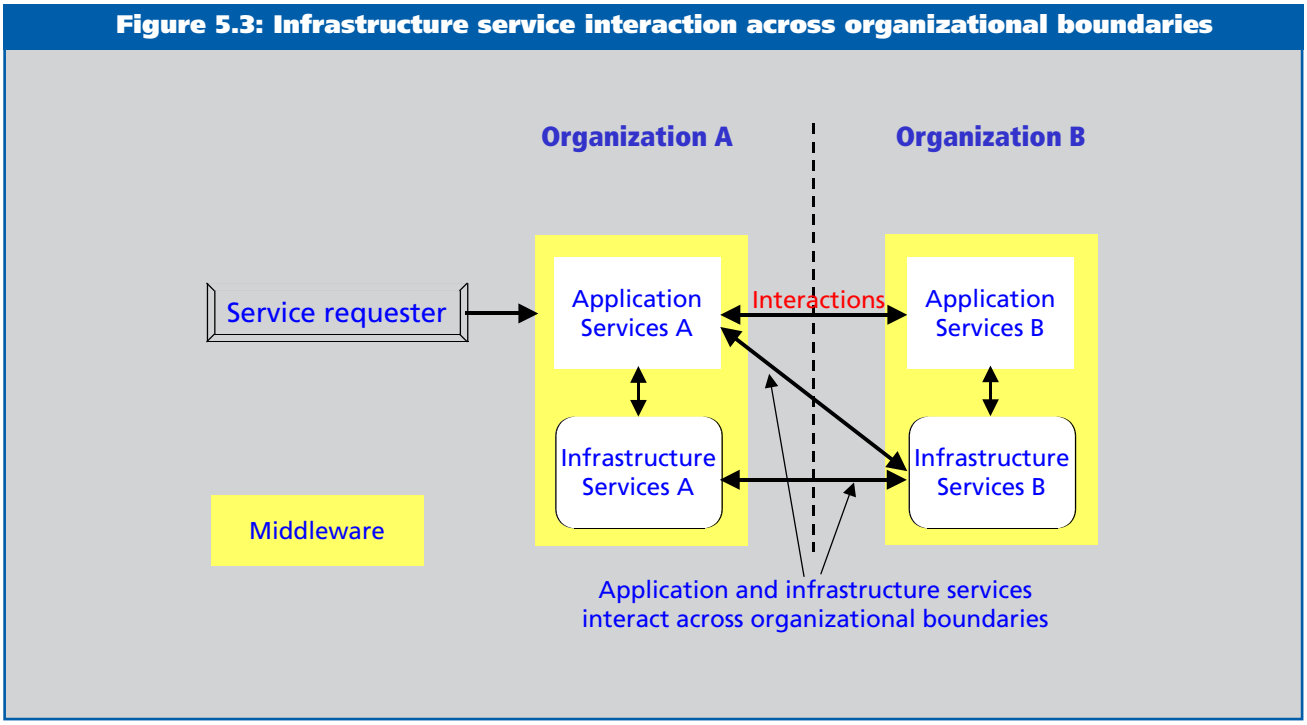
This suggests a requirement in the protocols for negotiating what the set of environments the resource provided can offer. The increasing use of Java and J2EE does simplify — by abstracting the application services from the underlying infrastructure, apart from the JVM and the application server.

The above two examples start to look like Grids. Using the checklist of three characteristics identified by Foster, they:

- **involve resource coordination across autonomous organisations, without overall, centralized resource management**
- **assume the protocols used are standardized, thereby meeting the second criterion**
- **might produce results that could be non-trivial (although this will depends on the application).**

The key difference between the second two examples, and the example shown in Figure 5.1 (which does not represent a Grid) is the first criterion: the ability to deliver resource coordination across organizations without a centralized control point.

Figure 5.3: Infrastructure service interaction across organizational boundaries



Architecture and implementation

As may be imagined, the services and protocols required to enable the successful construction of Grids are non-trivial, especially concerning the infrastructure services involved in distributed resource co-ordination. The fact that different organizations are collaborating — where each has its own systems administration, rules, regulations and diverse system types — means that standards have to be carefully defined to take care of the heterogeneity of the overall environment.

Conventionally this would be largely impossible. But the Internet has provided a wealth of experience in dealing with heterogeneous environments, comprising a mixture of autonomous organizations supplying networks and services of many kinds:

- **the development of the Internet protocol suite (also known as TCP/IP) has shown the value of standardization on protocols**
- **the addition of World Wide Web technology on top of the basic IP networks, including the more recent Web Services developments, has shown just how much more can be achieved.**

The Open Grid Services Architecture (OGSA) builds on IP network services to provide a variety of services for Grid applications. OGSA defines a service-orientated architecture to address common Grid requirements for systems management, collaborative computing and so on. OGSA uses Web Service standards, extending them if required.

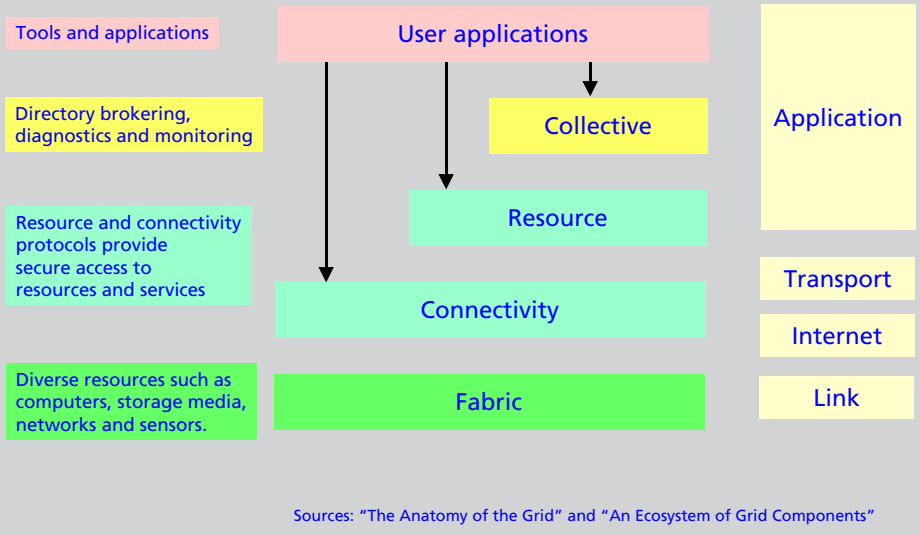
Figure 5.4 shows the architecture, as well as the functions of the layers and their relationship to the TCP/IP protocol stack. (*A more comprehensive introduction can be found in Foster, Kesselman and Tuecke 'The Anatomy of the Grid' — at http://www.globus.org/ogsa.*) In common with other layered architectures, each layer builds on the functions offered by lower layers.

The resource and connectivity layers provide a restricted set of protocols for access to a diverse set of resource types in the fabric layer and can be used to construct a wide range of services in the collective layer. The resource and connectivity layers can be seen as the neck of an hour glass, and are shown as such in 'An Ecosystem of Grid Components'.

The collective layer contains a variety of protocols and services for use with a virtual organization, including the following examples:

- **directory services, to discover the existence and properties of resources**
- **scheduling and brokering services, to request the allocation of one or more resources**

Figure 5.4: OGSA, showing the layer functions and the TCP/IP protocol stack



Sources: "The Anatomy of the Grid" and "An Ecosystem of Grid Components"

- **services to support the management of storage, networks and processing, to maximize performance**
- **workload management and collaboration frameworks, for the management of multi-step, asynchronous work flows**
- **security services**
- **accounting and billing services.**

At first the Grid community and the Web Services community worked without much co-ordination, although Grid implementations used existing Web technologies such as HTTP. Even so, the collective layer of the architecture seems to fit with the Processes layer of the Web Services stack (Figure 5.5).

This state of affairs did not last that long. Instead the two communities worked together subsequently to identify common requirements. In turn these have led to a set of specifications known as the WS-Resource Framework, or WSRF. WSRF specifies how to provide Grid implementations using Web Services implementations (again, see 'An Ecosystem of Grid Components' for more information).

Turn now to implementation. Early implementations showed that the software — the middleware — required to solve the problems in a particular project could in many cases be re-used elsewhere, although usually after some generalization. This was unsurprising. Those involved took the pragmatic approach of generalizing software from logic developed for specific cases. Indeed, much of the best software comes from solving a specific, real problem, generalizing and then applying the results elsewhere.

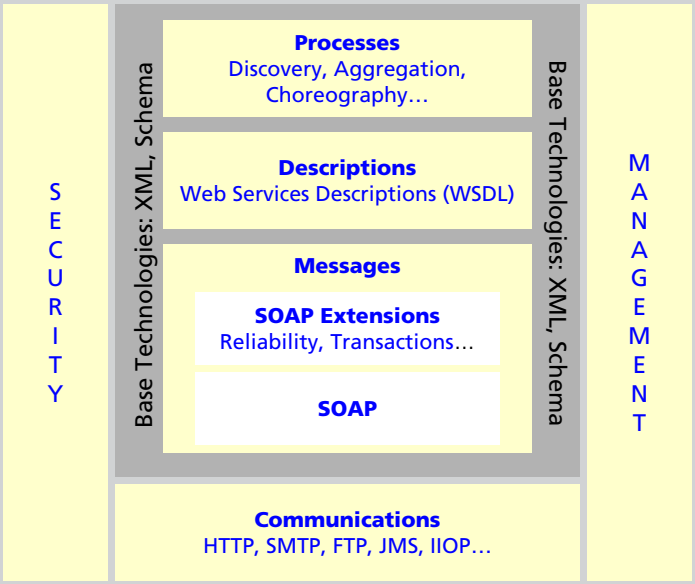
The result has been a steady expansion of generic solutions, which can be used by others and expanded. The Globus Alliance (http://www.globus.org/) has taken a leading role, providing an expanding toolkit for use in building Grids.

Grids as a computing utility

So far, Grids have been constructed to meet the specific requirements of various groups, creating virtual organizations. Yet there is continuing interest in the concept of a Grid as a computing utility, in a more global form. Indeed, the Economist published an article on this subject ("Who wants to buy a computon?" The Economist Technology Quarterly, March 12th — 18th, 2005).

So far there do appear to be significant problems of scale in achieving this. Grids assume that there is no central resource co-ordination. If this is the case, how does this tie up with determining priorities for work and pricing? If an organization has an urgent, high-priority requirement for infrastructure, it would probably be willing to pay more for

Figure 5.5: Web Services stack: the Processes layer positions the Collective layer



Source: W3C

the resources. A provider organization would then need an infrastructure able to suspend lower priority application services in favor of the higher priority activity, for which it could charge more. But how would it decide?

According to the Economist article, Hewlett-Packard is conducting research into a 'market-based' mechanism for priority and pricing management. The grounds for this are that a central planning scheme would collapse under the scale and that the analogy with a market economy could be useful, following a kind of stock market model. Users would be shown available resources, current workloads and prices. They would then bid for resources.

A final issue to consider in the utility model concerns accounting and billing for resources used. Electricity is sold by the kilowatt-hour. What unit is used for computing resource? Most operating systems have accounting features to charge for a combination of resources used. But in this case, the same operating system cannot be used. There needs, therefore, to be a standardization on concepts such as the job or task and the units of resource to be measured.

The HP research team mentioned above has suggested a unit called the computon, which combines various different resources — processor, storage and so on — into a single unit. The name is derived from the words 'computation' and 'photon'. Perhaps it is appropriate that the HP team have passed their work to CERN for more testing.

Management conclusion

In this analysis Mr. Bye has provided a commentary on Grids. He has discussed the origins of Grids, and the characteristics that a distributed environment should possess before it can be considered to be a Grid. As he points out, Grids enable a number of organizations to collaborate for some or all activities, and in so doing these form a virtual organization

He has also discussed the differences between a Grid and other distributed, inter-organizational co-operation. There are today many organizations co-operating across the Internet (or extranets), and increasingly using sophisticated Web Services. But, as he points out, these are not necessarily Grids. A key attribute of a Grid is that extensive co-ordination of resources is required, but there should be no central co-ordination of these resources, as there may be in a cluster.

Grids can be seen as a natural extension of distributed Web Services architectures. Indeed Grid and Web Services technologies are converging. Although much has been achieved, there is still a great deal to do before models such as a ubiquitous computing Grid are available. This means that it is still dangerous to hype the possibilities of Grid technology but, given what has been achieved with the Internet and its technologies in the past decade or so, there must be some confidence in the future of Grids. The fact that developments so far have been based on pragmatic approaches is encouraging.

Home
1. Industry
   1.1 WS-CHOR
   1.2 BPEL
   1.3 WS-Security
   1.4 WS-Trust
   1.5 Reliable Messaging
   1.6 WS-Transaction Framework
   1.7 WS-Resource Framework
   1.8 Web Services Composite Application Framework
   1.9 WS-Policy
   1.10 WS-Addressing
   1.11 WS-Federation
   1.12 WS-Inspection
   1.13 Resource Management
   1.14 WS-Manageability
   1.15 WS-Provisioning
   1.16 WSXL
   1.17 Security
   1.18 Microsoft
   1.19 WSLA(Web Service Level Agreements
   1.20 WS-MetadataExchange

2. JCP
3. OMA
4. IETF
5. Globus Alliance
6. WS-I
7. FIPA
8. OMG
9. OpenGIS
10. ITU-T
11. ISO/IEC JTC1
12. Advanced WS
13. BPMI
14. SWSI
15. GGF
16. Parlay
17. Related Specs
18. Liberty Alliance
19. OASIS
20. W3C

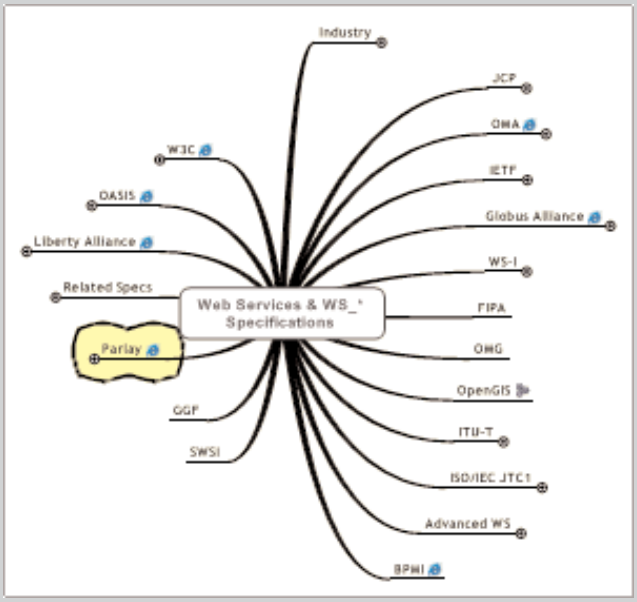


Figure 6.1: Web Services and WS-* Specifications (Source: http://www.w3c.or.kr/~hollobit/roadmap/ws-specs/)

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]