MIDDLEWARESPECTRA

incorporating FINANCIAL MIDDLEWARESPECTRA

Contents

February 2003

Volume 17 Report 1

Using EAI to improve existing processes

Mike Hastie
Director of Applied Research
Promenix

Management introduction

Promenix is a systems integration consulting company focused on assisting Fortune 500 Companies integrate disparate applications. This encompasses conventional back end, intra-company applications through front end, inter-company ones.

Promenix was founded in 1997 by four individuals who had been at 'Big 5' accounting firms and had seen the promise of messaging as well as its potential importance as a catalyst and facilitator for new types of solutions in an enterprise world. Mike Hastie was one of those principals; he is now the Director of Applied Research at Promenix where much of his work involves identifying and developing internal tooling initiatives, specifically what Promenix classifies as 'accelerators' for delivering application integration. These are designed to enable integration projects to move forward faster.

In this case study, Mr. Hastie discusses a project that Promenix undertook for a major industrial client in the business machines sector. As he describes, the project included working with a large SAP R/3 implementation team. The objective was to ensure that the R/3 application is integrated with existing applications. The software used for the integration infrastructure included what are now called WebSphere MQ (then, MQSeries) and WebSphere MQ Integrator (MQSeries Integrator). For reasons of commercial sensitivity, this industrial client prefers anonymity.

The client and proof of concept

Let me start with a short summary of what we were asked to do and what we did. Our client is a US-based $4.2 billion global messaging company — making copiers, faxes, mail sorters, postage meters and the like. Its mission, by its own description, is to provide to its customers the various products and services needed to support a message-based world. When we were retained, its management believed that one of the ways it could demonstrate leadership to its customers was by incorporating a message-based infrastructure into its own organization.

In August of 1998, Promenix was asked to assist with the creation of a proof of concept (PoC) project to demonstrate the use of IBM's MQSeries as a messaging backbone for application integration. If the PoC proved to be successful, it was intended to be used as the foundation for a message-based integration architecture for supporting several large packaged application implementations, both those scheduled for imminent implementation (specifically SAP's R/3) as well as any that might occur in the future.

When we started at this client we knew that the PoC project was vital if the move to a message-based infrastructure was to be credible. It was imperative that we demonstrate successes for each of several pre-defined PoC criteria while simultaneously making sure that our solutions were based upon a solid, dependable technology foundation which could subsequently be re-used in a production scenario.

The PoC activities, therefore, included:

- **developing the project plan**
- **designing solutions for the criteria**
- **installing and configuring software**
- **implementing solutions**
- **providing a management framework**
- **documenting the expected results against the actual results.**

We set up milestones for the project to show incremental achievements against the criteria on an ongoing basis. This provided the client's overall project manager with information regarding status, progress and issues during all phases of the PoC. It also provided us with a structured framework within which to overcome the various technical issues that arose.

The results of the PoC were positive. Promenix was then retained to build an enterprise-wide middleware-based solution to support the client's concurrent R/3 implementation. This included working with a newly formed internal middleware group which was responsible for architecting, designing and developing the message-based infrastructure by which it intended to integrate its existing and new applications. Besides the new R/3 applications, other applications included CICS, IMS, Siebel's CRM and several e-commerce/Web sites.

As I will describe later, the integration project envisioned the interconnection of more than 30 applications using over 100 interfaces. This was to be delivered via a messaging-based hub and spoke approach.

The project went live on schedule and runs without interruption of business. Volumes at each month-end now exceed 500,000 transactions per day.

Global messaging company

Now let me be more specific about what was achieved. The business of our client is innovative, world-class mailing systems. In essence, the business is oriented around the systems and services needed to process large mailings — including bulk and third party mailings. For example, your mail has a 'frank' or 'stamp' in the corner that says somebody has paid to send you that mail. Our client sells the hardware systems and the software — and, if appropriate, the services — that facilitate the production of large (or small) targeted mail shots.

The over-reaching objective that the client sought to satisfy was improved customer responsiveness. This was made more complex by its customer base — which comprised organizations of all sizes, not just large corporations. It mattered just as much that they could satisfy the needs of a mailing system or service for a small operation as for a multinational one.

Back in 1997/98 our client's management perceived the need to adjust to 'Internet realities' — expectations of near instant delivery as shaped by the likes of Amazon.com and Dell.com. Just as customers expect to be able to order books or computers online, businesses expect to be able to order, and then track, mailing services or machine delivery online.

Our client, therefore, saw the opportunity simultaneously to reduce inventories, shorten cycle times and increase customer participation and improve satisfaction. The intention was to integrate its applications so as to accelerate its business processes. The desire was to reduce the order delivery time from around six weeks to a time substantially less than that. So the objective we were dealing with was how to optimize the business cycle in such a way that our client's customer expectations might be satisfied.

Latency and error avoidance improve RoI

In brief, we addressed the batch processing issue. By doing this we were able to remove time from out of the existing cycle without having to modify the underlying business processes. Although not simple to deliver, the principles were straightforward. The main processing systems were largely, but by no means wholly, mainframe based — and many of the applications were highly efficient although batch-oriented. This imposed certain constraints.

For example, when an order arrived from the Web, it would take a day before the batch process was run that entered the order into the main order system. Another day would pass (until the next batch processing run) before the entered orders went into parts explosion. That parts explosion would take another day before being entered into an ordering plan or into manufacturing.

Then there were the delays caused by waiting for the delivery of components needed for some orders. When these items arrived, it took a day to be received into inventory. In effect, another day would pass before manufacturing would know that the required pieces (to start making the product) were now available. This continued all the way through shipping and accounting.

Most of the time delays occured in the waiting period for the next batch process; the processing between the various different applications occurred once a day, most commonly as some form of file transfer between each application. In effect, the more applications that needed to work together, the longer the inherent latency. The bottom line was that, if it took a day per business step (because of the batch approach), the company was prevented from being as responsive to its customers as it desired.

Another client objective concerned the improvement of customer satisfaction — whether reducing the numbers of errors, correcting ordering or shipping mistakes or responding to customer inquiries in a more timely manner. Our client's management had understood the implications of the Web: that people could (and would) order around the clock, internationally and domestically. But enabling orders to be accepted in this way was not a complete answer.

An example illustrates the challenge. Adding Web order entry had been straightforward. But customers frequently found that they mis-ordered — that they had forgotten an item or ordered too much. They would telephone our client's order call center, perhaps only 10 minutes or an hour after placing the original order, in order to amend that order. But the call center could not see those new order details — because it took some hours for the order system to populate our client's customer relationship management system. This was not good for customer relations, especially when the customer was referencing orders which the call center staff did not even know existed.

Our client basically said that it wanted its online systems to allow its customers to make changes over the Web — and that orders must be communicated to its own support staff as soon as possible. The same information that was available via the Web site to customers had to be available to the call center staff. The two systems needed integration so that each would be in synch. with each other, 24 hours a day.

While this may seem obvious in retrospect, it was not at the time. Nor was the additional benefit — that errors would be reduced and that customer satisfaction would soar. Similarly, integrating applications removed other areas of exposure, where re-keying errors frequently occurred and improved information consistency. Instances of having two customer records, because one was mis-typed, declined. In turn, this reduced the need for labor-intensive reconciliation and subsequent data re-entry or correction.

In effect, the client was saying to us that, if the online ordering system and the call center system and the back end systems could be synchronized, then the volume of errors would diminish. In turn this would improve customer satisfaction.

R/3 as a catalyst for messaging

The technical background at our client was typical of a large industrial organization of long standing. Many of the applications were running on mainframes under either CICS or IMS. In addition, other applications ran on the usual mix of UNIX and Windows-based systems.

As mentioned earlier, our involvement was not the only catalyst for change. Our client had already decided to implement SAP's R/3 in order to encourage change and to assist in the evolution away from batch-oriented processing. The main issues that remained were how R/3 was to be introduced and how it was going to be connected to all those business-specific applications that were not going to be replaced by R/3.

The customer clearly recognized that a messaging-based infrastructure was key in the integration of the new R/3 applications as well as the existing applications. One of the reasons that we believe the implementation was successful

was that the client's management, having made a choice to go with a messaging infrastructure, supported its use and implementation. For those people who did not want to change, who preferred to 'stay with what they knew', their resistance was 'broken' by a corporate mandate in favor of messaging. If you did not want to use messaging, you had to justify why not. Only then might you revert to traditional ways, if your arguments stood the test of peer review and applicability in a business context. (Few such objections were, in practice, sustained.)

Another factor that assisted us in convincing our client's IT staff that messaging was the right way forward was that the client had already invested considerably in Web support and Web Portals and the like, as well as CRM. It did not take much analysis to see that these needed to be connected to each other, and to R/3, if integration was to happen.

Trying to achieve this via (say) a synchronous mechanism was clearly going to be difficult, if not impractical. In contrast, messaging is not unlike batch processing, with each message being a batch of one — which is immediately processed rather than waiting for a specific time window.

The validity of the messaging concept was also borne out by the number of interface points that needed addressing. There were, in Phase I alone, some 35 different integration points, with at least 100+ different interfaces (more were added later). All these needed to be addressed for it was clear that R/3 was not going to satisfy all requirements.

Although the intention was that, in subsequent Phases, the number of integration points would diminish, both our client and our own thinking acknowledged an unwelcome reality — that 'temporary' interfaces rarely remain temporary. They become established.

A further confirmation that the asynchronous nature of messaging was appropriate came from acceptance that not every application would be replaced by R/3. This meant that a 'hybrid' state — R/3 working with other applications, including new ones as well as existing ones — was going to exist into the foreseeable future. Flexibility mattered. Messaging could provide it.

Understanding this enabled us to introduce the need to put 'best of breed' in context. Best of breed used to be an ugly description, because of the cost associated with it.

But best of breed, when it exploits messaging to enable two different systems (that do not really know that they are talking to something different), makes for a significantly cheaper solution. That is not to say that there is no cost associated (there is, for the messaging infrastructure and management) but the application function — which is what usually matters most to the business — can remain as best of breed.

The value of a messaging hub

When we tied all this together, as part of the infrastructure for our client, it became clear that a 'messaging solution' encompassed more than just messaging. It needed an infrastructure that was oriented to — or designed from inception for — moving data between multiple applications with automated format transformation and message routing.

In this context we recommended that, if the client was going to 'bet the farm' on the messaging approach, common sense said that the selected software had to be stable and reliable — and come from a well established organization that was not likely to provide surprises six months down the line. In the messaging world that left relatively few products and vendors that could deliver (and none from the JMS world, which was probably the most logical 'standard' alternative).

IBM was already a major provider of systems and software to our client. The choice of MQSeries (now WMQ) was, therefore, fairly straightforward — especially as IBM products have support, maintenance and long-term strategic direction behind them. In proceeding down this road our client was doing what we see more and more companies doing today — adopting solutions that are proven and sourced from an established, reliable vendor.

For the R/3 implementation, our client had already decided to go with eight top-of-the-line AIX boxes, and that was only for production: development and testing were in addition. But these R/3 images needed to be connected to all the other applications. So the client chose to purchase, in bulk, enterprise-wide MQSeries licenses. An MQSeries instance could then be placed on every application system that needed to talk to another application. This, then, provided the raw messaging infrastructure.

On our advice, the client went further. As I have described, a key issue was that the Web-supporting applications needed to talk to the ordering application, and the ordering one to the CRM application as well as to inventory control and to manufacturing and to shipping, etc. There was a potential choice here — to go for a series of point to point connections provided over MQSeries or to choose a messaging hub.

The difficulty of the point-to-point approach was that complexity increases when n (the number of applications or interfaces needing to communicate) grows larger using an n(n-1) connection algorithm. Once **n > 10**, the management/maintenance burden becomes significant. In this case, there were a minimum of >30 applications in Phase I alone.

The advantage of a hub approach came in two forms. The first was of simplification, from **n(n-1)** to **n+1**. Every application would have a (two-way) MQSeries connection to MQSeries running on the hub. A message, carrying data or an event, would go from Application A to the hub where it would be routed to the appropriate destination, or destinations. This latter capability introduced the second advantage — that a message arriving at the hub would exploit the hub's own logic to transform the message into the format that any recipient application(s) needed; the hub would then route these transformed messages to as many destination systems as were needed. Thus an order coming in from the Web would go to the hub where it would then be routed to the order entry application and the customer support application as well as any other destination required.

This was much simpler in concept than having to write individual interfaces from the Web order entry application to the main order application, the customer relationship application and each and every subsequent application. Furthermore, if a source or destination application had to be added, it could be, without having to rework everything that was already working.

We recommended that our client implement MQSeries Integrator (now WebSphere MQ Integrator or WMQI) as the hub. This sits on top of the MQSeries layer and provides 'EAI Services' (Figure 1.1). It can:

- **read messages off inbound queues**
- **process these through pre-established transformation and routing logic**
- **send out the resulting messages to the designated destination(s).**

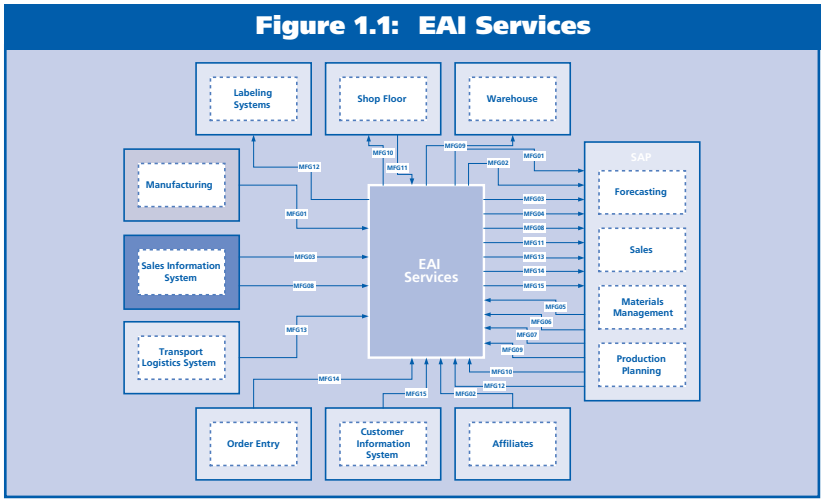
Architecture and practicalities

By using WMQI, our client believed that a hub and spoke architecture (with the applications at the end of the 'spokes') was the way to progress forward. A messaging hub provides a point of concentration for transformation operations and for locating the different routing required. In the case of the transformations, the format for an incoming message had only to be stored once, as did the format for an outgoing message — for each application. The advantage was these formats needed to be defined only once. When in place, a message from application A (to the hub) could be matched (at the hub) to the outgoing format that Applications B or C or D required. In this way a great deal of duplication of effort (that would have been required in a point to point mesh) was avoided.

That is not to say that a hub and spoke approach is without additional costs. A hub is a concentration point. It becomes central to business activities, especially when conveying messages between different applications.

We had to design the WMQI hub, and its associated system and management software, on the principle that such a hub is mission critical and must always be available — or possess appropriate fallback and failover hardware, software, and procedures. Nevertheless, the cost of implementing high availability was substantially less than the point to point approach.

Figure 1.1: EAI Services



A different practical issue that continued through Phase I was the inclination of many of our client's IT staff to 'prefer' to go for individual point to point solutions, even after we had implemented the WMQI hub. This was when we had to exploit management's mandate and extend this to cover the hub. Developers were not permitted to use a point to point messaging solution unless they could demonstrate why the hub approach was less efficient for the business.

During this process, an interesting discovery was made. We were constantly sur-

prised by the number of IT people who thought that 'their' data was never, or should never be, used by others.

Experience demonstrated that the reverse was more often true — that others could use the data if only they could obtain access. This the WMQI hub enabled, in ways that had previously been impossible. In so doing, the duplication of data and errors that I discussed earlier was reduced further — all contributing to the speeding up of business processes as well as the reduction of internal inefficiency. At the end of Phase I, we had 35 applications hooked to the hub, with 130+ interfaces. This took about nine months in total. Furthermore, as more and more interfaces were delivered (Figure 1.2), progress accelerated and became easier and easier (because so much had been done already that could be re-used).

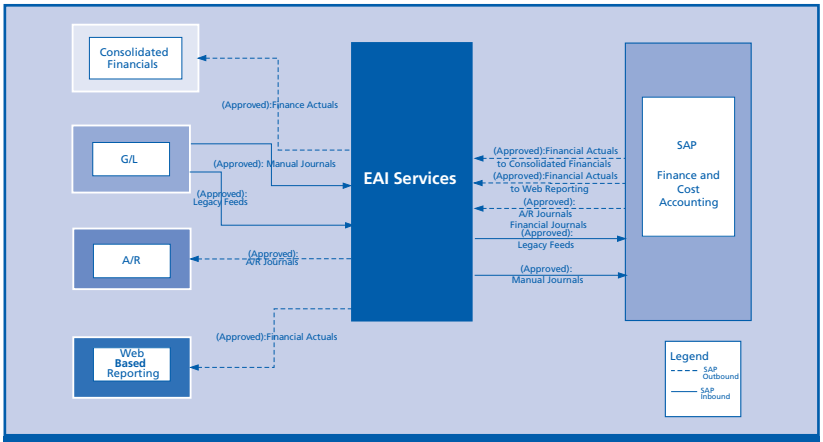


Figure 1.2: Consolidating applications

In contrast, if we had gone down the point to point route:

- **the time needed would have been much longer with much greater expense**
- **each point to point interface would likely not have been re-usable in the ways that WMQI enables.**

Lessons learned

Lesson one was — if you can do what you already do better, do it. It is easier to adapt existing processes as well as being much cheaper than trying to replace or re-engineer proven processes (and applications). Interestingly, this had immediate RoI for the Company. It was much less expensive, and faster to deliver, than formal business process re-engineering.

A significant second lesson was that major change can be delivered if the right support is in place. We had a lot of resistance in the beginning. 'It is just not going to work for my systems' or 'that is just not a reasonable thing to ask us to do given the project deadlines' were common statements. We, both our client and ourselves working together, accomplished what was required in the timeframe set. The point to take away is that you can introduce significant technological change with the right planning, design and approach.

Associated with this is the importance of Proof of Concept projects. Proving a technological approach works makes it much easier to persuade business or IT people to switch. Obtaining a corporate endorsement, preferably at the highest level, is hugely helpful. At our client, IT people really could not say no. They had to explain why they wished to push back. Once they laid out their logic, it was easy to demonstrate that the messaging alternative was equally as good and usually better than any other approach.

My final lesson confirmed the value of co-locating the application and system and middleware people. This does not sound all that significant. But it was. We were all able to walk down a corridor and interact with our peers. This reduced the delays and iterations that experience has taught Promenix are endemic when teams are geographically dispersed and with a minimum of informal interaction. I personally have been on projects where one part of a team is in Europe and another in the US. You can lose days even though the rapport is good between team members. This did not happen with the client — and it definitely made the project delivery both simpler and faster.

Management conclusion

In terms of the improving business processes, Promenix illustrates the relevance of process built upon messaging, transaction, development and production. Perhaps most importantly, its (Promenix's) client has been able to refresh its existing batch-oriented processes and make them much more responsive to what the customer needs — without having to rewrite them. At the same time, where new applications were needed (like SAP's R/3), these could be accommodated.

The result has been an acceleration in the time to complete start-to-finish business processes, pleasing customers and management alike.

Many Grids to fit many tastes and requirements

Jim Gray
Distinguished Engineer and Manager
Microsoft Bay Area Research Center

Management introduction

Jim Gray is a Microsoft Distinguished Engineer. He is part of Microsoft's research group and is Manager of the Microsoft Bay Area Research Center. Over many years his work has focused on databases and transaction processing and he was awarded the ACM Turing Award for his work on transaction processing. He has also been active in building online databases, like those found at ***http://terraService.Net*** *and* ***http://skyserver.sdss.org****.*

In this discussion (which first appeared in **GRID.MIDDLEWARESPECTRA***, downloadable from* ***www.grid.middlewarespectra.com****), Dr. Gray talks about his view of the Grid and its link to middleware. He describes:*

- ***how 'the Grid' is composed of multiple communities and interests***
- ***the challenges that face each***
- ***Web Services and OGSA, and places these in context***

- ***the commercial dimension.***

An informal Grid taxonomy

The Grid has at least five identifiable communities, each with a different interpretation of what it (the Grid) is about. These include:

- **compute-centric users**
- **peer to peer advocates**
- **outsourcers**
- **data and application-centric advocates**
- **collaboration-centric (for example, Access Grid) proponents.**

The term 'Grid' was coined by compute-centric users led by Ian Foster and Carl Kesselman. Most of these people come from US government funded laboratories and supercomputer centers like those at Urbana, Argonne, Fermi, San Diego, Livermore and so on. The original Grid book *(I. Foster and C. Kesselman [Eds]. The Grid: Blueprint for a New Computing Infrastructure — Morgan Kaufmann, 1999*) had chapters about collaboration, data, computation and instrumentation. This community certainly understands the whole Grid spectrum and it is actively involved in projects that encompass data, telepresence and portals. Nevertheless, much of the discussion here, and the work actually performed to date, focuses on running large computations (resource management) and moving the input and output files about (GridFTP).

This community typically runs big MPI batch programs. They operate huge machine clusters costing millions of dollars. Most of this class of user want access to some supercomputing somewhere. In addition, some of the participants need to perform computations that are even larger than the available clusters; they need to connect these supercomputer centers on a planetary scale. In this context, they envision the Grid as the sharing of compute resources.

One of the paradoxes of the supercomputing world is that SETI@Home delivers about 45 tera-flops of processing, all day every day. This is 30% more than the peak performance of the NEC Earth Simulator that claims to be the biggest, fastest computer around today. Yet, SETI@Home is more powerful than the bottom half of the Top 500 combined.

My point is that, if you want CPU cycles, most of them are not in data centers, they are on desktops. Furthermore, Condor, Entropia, United Devices and others are harvesting these 'spare' cycles using peer to peer technologies — rather than the centralized computer-center model.

These peer to peer advocates form the second active and innovative branch of the Grid community. Broadly, they have a Napster, SETI@Home or Gnutella heritage. As such, the peer to peer community wants to program the Web as a symmetric system of autonomous nodes (as opposed to a client-server architecture). This model puts the processing at the periphery of the network. In addition it places most of its storage at the periphery and it is looking for applications that can harvest the capacity at the edge of network where the nodes operate under decentralized control and interact via symmetric protocols (rather than in a master-slave relationship.)

Moving on, the third group comprises outsourcers. Many have observed that the Grid is a resource. You send your problem to the resource; the resource sends back the answer. The biggest outsourcing company in the world is IBM. IBM Global Services is embracing and extending the Grid concept. IBM is one of the leaders in the Open Grid Service Architecture (OGSA) as a natural evolution of its outsourcing business. Other outsourcing companies are also climbing on this bandwagon.

Next there is the 'application and data centric' Grid subculture (and I count myself in this group). These include people who believe that harvesting CPU cycles is not very interesting. That is not to say that compute intensive jobs do not exist; but most of these tend to be specialized in nature (like SETI@Home).

Instead, this application and data centric group is more concerned about using applications and their data. Questions are sent to the application/data servers either because the data is huge (petabytes) or because it is encapsulated (proprietary). Application servers provide answers, with portals integrating answers coming from multiple sources.

This application and data centric group believes that what we are trending towards is a world in which:

- **there are data centers, not computer centers**
- **most of the costs now lie in data management and networking (while computing costs are less than sales tax)**
- **you send requests to the data centers**
- **when requests arrive at multiple data centers, they face an interoperability problem.**

To make everything work as desired, we need to have ways for:

- **data to travel from one place to another**

- **intermediate results to travel from one place to another to be composed into answers.**

This application Grid model is much more data-centric than compute-centric. It embraces all the OGSA, Web Services, .NET, SunOne, peer to peer technologies and anything else that works. Its goal is to make it easy to:

- **program the Web , in order to access the data in it**
- **send not just bytes but information (objects and the methods associated with objects) around networks.**

The current plan is to build the Data Grid using OGSA which is in turn based on Web Services, XML and SOAP. What is significant here is that SOAP really is a Simple Object Access Protocol. The layers that are built above it really are an object model. You could say this is 'Internet-scale CORBA' that really works.

The collaboration-centric 'Access Grid' community forms my fifth Grid community. It wants to use the Grid's network bandwidth to allow high-quality person to person communication. As such the Access Grid is a teleconferencing and distributed meeting software platform that enables people to minimize travel while still communicating effectively. To achieve this, the Access Grid needs copious bandwidth which means it needs a very high speed network — which is one deliverable that the Grid is likely to put in place. Indeed, my colleague Gordon Bell asserts that the Access Grid will be the most significant out-growth of the Grid, much as Mosaic (the browser) was the main social benefit of the previous supercomputing generation.

Yet, multifaceted 'interpretations' of the Grid are good; there is something in it for everyone. If they look, most people will find something in it that they really like about the Grid. The Grid is a big tent in which everyone can find their dreams.

The major challenges and reality

The major challenges depend on where in the big tent you happen to be standing. The Access Grid is a fascinating research area. Right now the software and hardware base is chauffeur driven: you have a special room managed by an Access Grid specialist (who needs to set up the four different linkages and operate the four different computers and many different applications).

If you participate in an Access Grid event there is a good chance that some piece of the technology will not work at the right moment. As such, the Access Grid is fragile and its architecture needs two more prototype generations before it will become useful to you and me.

Nevertheless, I think the probability of success of the Access Grid is certain. This is why Gordon Bell is so enthusiastic about it. Microsoft's ConferenceXP is our shot at the next generation and is receiving good reviews — it is easy to use and can be inexpensively set up in your office or in a small conference room. It uses one computer, lower bandwidth and represents a lower cost version of the Access Grid. The office/laptop version runs on your current system. A classroom is about US$25,000 and is self-managing. We are now working actively on trying to bring down both the cost and the complexity of having an Access Grid node.

The peer to peer guys continue to have just extraordinary enthusiasm. In part I think this is because their approach appeals to people's democratic and egalitarian notions- where everybody can contribute to the computer in the sky. Napster was as much a social movement as anything else: students had Napster decals on their dorm room windows.

To me, the peer to peer community is robust. The challenge it faces, however, is finding applications. File sharing was one application — sharing of music or recipes and so on. SETI@Home is also a kind of peer to peer application and there are companies founded to ride the peer to peer revolution (although most of those companies are now gone).

The peer to peer space is, therefore, competitive. It is not an easy space in which to establish a profitable business model. Both United Devices and Entropia (for example) have found their best customers to be large organizations that want to harvest the spare cycles within their own enterprise.

This 'within-the-enterprise' cycle harvesting is a pretty narrow form of peer to peer. But it is still relevant. One observation that a customer made to me recently was that the likes of the US Government, AT&T, Ford and GM already have substantial internal networks. In fact they are provisioning their networks with gigabit backbones. In practice, they are already in a position to build 'intra-Grids' within their organizations. Indeed, I believe that United Devices and Entropia are having their biggest successes by building such intra-Grids for harvesting these spare CPU cycles and supplying these to others within the same organization who are cycle-limited (rather than I/O limited).

The peer to peer guys are now coming out of the manic phase when there was such incredible enthusiasm and

hype. They are now having to deliver. The challenge they face is finding more applications that can exploit their vision for edge nodes on the Internet and Intranets.

So, moving on to the next Grid group, the application and data centric group is the area where I have been working. It has a great deal of velocity. In Microsoft, the .NET guys are beavering away. There are several thousand developers in Redmond actively programming to build .NET and really exploiting the emerging GXA stack of advanced Web Services. There are similar efforts at IBM, Oracle, BEA and other large software houses.

And, of course, there are the Physics, Biology, Medical, Earth Sciences, Chemistry and other scientific disciplines that are embracing the Grid as a way to publish and access scientific data. This is happening in Asia and Europe, as well as the Americas.

This is also the area which is most familiar to those interested in middleware, including traditional IT. It is where conventional data processing meets the Grid and addresses — naming, security, authorization, data representation and interchange, programming models, transactions, co-ordination, work flow, etc.

To give some idea of progress, there are about 25 Web Services specifications in the pipeline, having to do with anything from queuing to work flow to how to represent data, schemas, query languages, etc. (Microsoft calls this GXA for Global XML Web Services Architecture.) The volume of activity is immense and, speaking personally, it is almost impossible to stay current with all the things that are going on. There is just so much happening.

I know this area is Microsoft's main focus, and I believe other software companies are moving in the same direction. Sun's recent joining of Web Services Interoperability (WS-I) seems to have ended the final hold-out.

It is easy, however, to discount this, to say 'I've seen this phenomenon before, this is OSI or ... all over again', where many people met to agree standards but few were interested in implementing many of them. I think the Grid really is different, but only time will tell. I see two major differences. The first difference is that people are implementing in parallel to the definition of the specifications. Some leading-edge customers are already in production: that is different from OSI (for example).

The second major difference (from past efforts) is that the specs. are pretty straightforward. They start with issues that people care about — like how to:

- **represent data**
- **present queries**
- **provide security.**

.NET was a huge gamble on Microsoft's part. It thought that XML was different. Around 1997, it redirected a large part of the company towards XML Web Services. .NET came out around 2000, after three years of engineering. Today we are five years down the road and it looks like a pretty good gamble. The parts are now coming together.

Let me illustrate why with a story. I was at a customer event recently. There was a panel about Web Services. The panel was chaired by a pundit who started by saying that Web Services were 'just old wine in a new bottle', that there was 'nothing new or good here' and that it was 'all hype'.

The next speaker was the CIO who said 'Well, yes and no. My company is a J2EE enclave, using WebSphere and WebLogic. We could not get WebSphere and WebLogic elements to talk to each other through RMI. So, rather than trying to talk in the same language (Java), we decided to talk in a different language — Web Services. It worked. Yet our reality is that it remains difficult to get two different RMI implementations to talk to each other.'

This CIO went on to describe how his organization had some ten major projects under way using Web Services. Three of these had already been deployed and two were in beta. The resultant systems are much more maintainable, much less fragile (than the binary based systems they had worked with before), primarily because his people could actually look at what was crossing the wire with tools that already exist.

Web Services really do focus on interoperability. They pay a huge price for going through XML, but the benefit of this is interoperability. Sending you a floating point number through XML is not pretty — but it is easy to make work at both ends of the wire. The relevance to the Grid is immense.

That said, there has been a huge amount of hype about Web Services. I suspect that there are a fair number of people who have no idea what they are. When they find out, they will be disappointed that they are so simple. For middleware experts, Web Services can be described as:

- **'XML meets CICS'**
- **or 'XML meets LU6.2'**
- **or 'XML meets CORBA'.**

That was what the pundit meant about putting new wine in an old bottle.

So the challenge here, and this is really a challenge for both IBM and Microsoft, is how seriously each company will take the Web Services Interoperability consortia (WS-I). Interoperability is critical. The need is for everyone to agree that interoperability is more important than going out to build a better system than is available from competitors. The fact that Sun finally joined WS-I suggests that no-one can now ignore it.

This brings me back to the Grid. The Grid may yet turn out to be a force for good — by keeping vendors true to interoperability as the critical need. In addition, the outsourcers will need this if they are to have a business.

Now let me turn to the progenitors of the Grid-the compute-centric people. Everybody is hopping onto the Grid bandwagon. But what is worse for this group is that computing is becoming free. It is much easier to deal with your local Beowolf cluster than it is to deal with some remote computer center. The reason people built Beowulf clusters is they could not stand to deal with supercomputer centers. In the end somebody has to pay.

The economic model for the traditional super computer group no longer works. It is too painful and there is too much overhead associated with getting CPU cycles from some high-overhead computer center. The only reason to go to a remote facility is that it may have something you want (like data or information).

That said, the compute-centric Grid community is working to build batch job schedulers to run huge MPI programs on large clusters. This seems somewhat retro to those of us who have spent our careers replacing:

- **batch programming with timesharing, and**
- **batch transaction processing with online transaction processing, and**
- **batch data analysis with online analytical processing.**

But, I guess batch is no longer retro. It is the wave of the future.

This Internet scale Beowulf faces a problem though: it requires problems that have huge instruction density. Indeed, you can characterize a compute job by:

- **how much I/O it does**
- **how many instructions it uses**
- **how much network traffic it requires.**

Right now a dollar buys:

- **1GB of network traffic**
- **one day of CPU time**
- **10 days of disk arm time.**

If you look at SETI@Home or problems involving protein folding, they do about a million instructions per byte of data. Given that (high) instruction density, it is economic to send the data elsewhere for processing (buying network time is cheaper than buying the processors and performing the job locally).

But most of the jobs I see have an instruction density of 10 instructions per byte to 10,000 instructions per byte. In those computations it is much cheaper to send the program to the data. For those jobs the CPU costs a tiny fraction of the data access and network transport cost.

So I think that the compute-centric guys need to move towards the data-centric model. This is not a huge issue; all they have to do is embrace and extend the data center model and they will be done. Indeed that is what OGSA is about. But it does move the focus away from batch job scheduling.

OGSA and Web Services

Right now OGSA is mostly wrapping Globus in Web Services — translating each Globus service to an XML/SOAP equivalent. This is a good first step. But the question that this raises is: what are the Web Services that arise when you have a strong object and data model?

The compute-centric folks are very bright and very hard-working. They are genuinely trying to make progress. I believe they recognize the data modeling and object modeling problems and are working hard to address them.

Paradoxically Tony Storey (of IBM) and Malcolm Atkinson (of Edinburgh University) — as well as various other people in the European Data Grid — are much stronger in the area of data representation. Something I find a little surprising is that Europe is likely to lead the definition of:

- **what a data set is**
- **how data is represented**
- **how XML data fits with the Grid.**

The reason for this is, however, understandable. Europe's financial support system is different to that in the US. Tony Hey in the UK,, for example, is concentrating on e-Science and e-commerce. Necessarily, that makes the UK's Grid activities much more focused on data and business possibilities.

Commerce and the Grid

Any large organization that has inter-operability problems is going to love Web Services. Web Services are becoming the base for the data-centric view of the Grid. Ergo, commerce should love the Grid — even if it has yet to discover this, and especially as the Grid logically extends the inter-operability available within an enterprise to working between two or more enterprises.

That is not to say that all is done. Web Services today are mostly about inter-operation between different parts of one organization. They usually do not require high performance, although high-performance Web Services will come with time and maturity.

While it is fair to say that XML performance remains dismal today, this is little different to the early days of HTML. There are already efforts to deliver high-performance XML at Microsoft, and it is not alone in working on this. The Microsoft model is that we will have proprietary implementations of these open standards. People will buy one implementation over another based on Total Cost of Ownership (TCO) — productivity, functionality and performance. This should be familiar: it is what distinguishes the various SQL systems today. For example, those people now doing TPC benchmarks using .NET. tpcC benchmarks with Web Services (rather than HTTP/COM+) are showing significant signs of improved performance.

If you want a different context, think of astronomers. I have been working on building a worldwide telescope. In the past astronomers had lots of telescopes and lots of data coming from those telescopes but few means to pull the data together in meaningful ways. That has changed. What was hard before .NET (and Web Services) is easy now. Most of this has been about solving inter-operability issues — essentially business to business communication or, in this case, astronomer to astronomer communication.

So I see the Grid as an engine of change, even of discontinuity. This is complicated. You can emphasize the similarities or the differences. The Grid will consist of at least my five 'views'. If you throw .NET and Web Services into the Grid, it is even more significant than it used to be. But what really matters for business is that it (business) adopts a view (of the Grid) that is most appropriate to its needs. Each business needs to look at the Grid's many aspects and look for what parts of the pie will help — rather than just blindly embracing peer to peer or Globus or an Access Grid because everyone else is doing it (and I do see a lot of this).

That said, there is a business dimension that I do not think many have considered. Much of the Grid discussion occurs in an economic vacuum — everything is assumed to be free. While Web Service applications do not need very much bandwidth, many of the Grid applications seem to need huge bandwidth capacity — because they are moving data around the network.

The problem is that such bandwidth is not there at an affordable price. The instruction density has to be at least a million instructions per byte of network traffic given current economic pricing. There are very few applications that match this profile. This is a real barrier.

In addition, there really are cultural differences between scientific application people and business applications people. We are trying to bridge these barriers. But this is difficult when the scientific guys fundamentally do not have the experience with structured data that they have with the MPI computational model. They remain much more oriented to files than databases.

Nevertheless, I think many of the tools out there are applicable to the various individual views of Grid computing. These are going to change many aspects of computing. For example:

- **peer to peer is finding niches in Monte Carlo simulations,**
- **the uptake of .NET and Web Services for building the data Grid is huge**
- **the Access Grid, if Gordon Bell is right, will likely overshadow all these developments.**

Management conclusion

Dr. Gray has been at the forefront of both transaction processing and databases advances over the past 30 years. He is in an excellent position to observe the evolution of the Grid, and to categorize its many interested parties.

In his discussion, three points stand out:

- ***the importance of Web Services, however named***
- ***the reality that the Grid is already supporting different interests with different objectives***
- ***the likelihood that the Grid will change the commercial world's way of operating, once it realizes what the Grid can deliver.***

Enterprise Application Integration — in transition

Steve Craggs
Principal
Saint Consulting

Management introduction

Inevitably, the EAI market has changed since its 'arrival' in the early 1990s. As often happens in new market segments where there is rapid growth and the emergence of numerous software vendors eager to seize the best possible share, the market has suffered from high degrees of hype and manipulation. New concepts and ideas have been propounded with increasing speed by vendors and analysts keen to raise their profile in this attractive market, sometimes with scant regard for users' actual needs. The resulting confusion has caused many companies to stumble blindly in a cloud of marketing and sales activity, having little clear idea about the difference between pragmatism and idealism.

This analysis, by Steve Craggs, seeks to:

- ***disperse some of those clouds of confusion***
- ***reveal a more accurate and practical picture of what is happening in the EAI market today***
- ***assess what is likely to change over the next few years.***

Background and evolution of EAI

Ever since its inception in the mid-1990s, the Enterprise Application Integration (EAI) market has seen dramatic growth as more and more companies start to understand and appreciate the value of delivering improved integration throughout their business operations. The concept behind it is essentially very simple — by improving communications between different applications, systems and environments it becomes possible to achieve greater levels of automation, efficiency and quality of service throughout the fundamental processes driving each business.

This concept has stood the test of the last five years or so, through some dramatic swings in macro-economic circumstances. During the late 1990s, for instance, one attraction of improved integration was the capability to address e-business needs rapidly, especially where connectivity between legacy systems and newly developed Internet-based components was a key requirement. Another attraction was the ability to introduce off-the-shelf packages within an existing legacy IT infrastructure.

As the millennium passed, the widespread economic downturn forced more and more companies to focus on the bottom line rather than on new developments. This did not, however, negatively impact EAI. Rather it provided a new driver, but this time for improved integration. Now the attraction lay in improving the return on existing investments — rather than supporting the delivery of new ones. Put another way, streamlining and automating existing business processes, both internally and across the entire corporate value chain, offered a reduction in operating expenses and improved profit margins. It is no coincidence that, even over the last couple of years, enhanced integration remains at the top of many company IT priority lists.

The EAI market develops

One aspect that has changed drastically over the last few years is the definition of the EAI market itself. Following the development of asynchronous message-oriented middleware (MOM) in the early 1990s, which made communications both easy and decoupled between heterogeneous platforms and applications, technology was added on top of the MOM to enable packaged applications from vendors like SAP and PeopleSoft to link more easily with legacy and newly developed Web and workstation-based applications. This new technology integrated applications in an enterprise together — that is, it offered 'Enterprise Application Integration'.

At this initial stage, EAI consisted primarily of one or more of the following components:

- **a message-oriented communications pipe, to provide the mechanics for connecting differing components**
- **adapters and/or connectors, to hook into the source and target application components**
- **a transformation engine, to map information from the format used by a source to the format used by the target(s)**
- **some form of routing software, to automate the delivery location of the next target.**

In a sense, it is not unreasonable to think of the advent of MOM as being Phase 0 of the EAI marketplace. Phase 1 arrived with the delivery of what became known as message brokers, that is products which offered the type of function listed above. For Phase 1:

- **some vendors built their own MOMs as part of their EAI offerings**
- **others decided to use one of the MOM offerings already available in the marketplace.**

During these early days of EAI, vendors prospered extravagantly due to the fact that the most common form of sale was to provide an EAI infrastructure within a company. This generally represented a substantial investment both in licence fees and services. Prices were generally high because of the small number of competitors. However, it did not take that long before a number of major transitions in the marketplace occurred. In hindsight, these can be attributed to:

- **competition became intense: more vendors entered the space, R&D as well as Sales and Marketing budgets rocketed and the market started to shake out with numerous acquisitions as a major battle ensued to determine the market 'gorillas'; interestingly, even now there is no one single EAI leader, with three companies fighting it out for the top spot — IBM, TIBCO and webMethods**
- **more and more IT budgets were moved under end user control; as economic conditions tightened, budgetary control was given to business teams who were expected to establish return-on-investment cases based on real business benefits — which meant that some organizations were purchasing several different flavors of EAI according to each business unit's unique need.**

At the same time, most EAI vendors were quick to realize that, in order to sustain revenue growth, it was going to be

necessary to climb the value stack as quickly as possible in order to differentiate each offering from those of the competition. The focus was on moving to a sales proposition which emphasized business-oriented value more than middleware and technology. All of a sudden the talk in the marketplace was about business process integration/management/automation (BPI/BPM/BPA), where the story was one about business process flowcharts created by business analysts who somehow mapped these process descriptions down to underlying IT components and programs.

While there has been wide spread vendor understanding of the significance of this new area of focus, not all have understood equally. The best do seem to understand what business processes are about; but others are clearly just 'putting lipstick on the pig', making cosmetic changes in order to be able to claim some sort of business process functionality.

The other main cause of transition in the EAI market was driven not by the vendors but by users — specifically those people who actually had to implement EAI solutions and move them into production. While the vendors were desperately trying to climb the value stack to claim the marketing high-ground, these implementers were discovering that there were major holes in the product offerings in the area of 'real-world' production usage. It emerged that, for example, most EAI vendors had paid little or no attention to migration and co-existence requirements.

The apparent assumption (on the vendors' part) was that once everything was engineered to use the new EAI technology then life would be fine. In reality most user organizations could not afford a big bang or clean sweep approach to any new solution, if only because the business risk was too great. Users decided that the old must live with the new, which tripped up those (many) EAI vendors that were not ready for such co-existence.

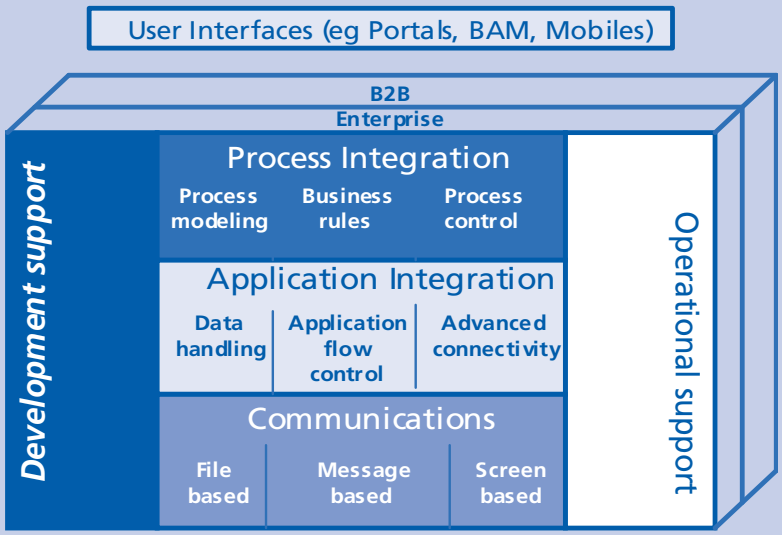
In addition, it also emerged that the support offered by vendors for production was patchy at best. Examples of this included the scarce consideration given to older, but proven, means of integration such as file transfer, screen-scraping or data integration. Furthermore, the sort of production tools required by most user installations, such as monitoring tools, were 'considered' by most EAI vendors as an afterthought, with only one or two notable exceptions.

Consequences and taxonomy

The result of all these different influences on the EAI market can been seen in the modern view of the marketplace laid out in the taxonomy in Figure 3.1. Companies that want, or wanted, to implement an EAI strategy often end up having to consider at least each aspect of this taxonomy in order to determine their best approach.

The disadvantage is that embracing so much often means working with more than one vendor. For instance, most

Figure 3.1: Taxonomy of EAI



EAI vendors remain reluctant to invest their resources in trying to address the bottom layer. They prefer instead to leave this to the system integrator or the in-house implementer — or to use partner offerings. It is the higher layers that receive most of the EAI vendors' attention due to the apparent improved linkage of these layers to business value.

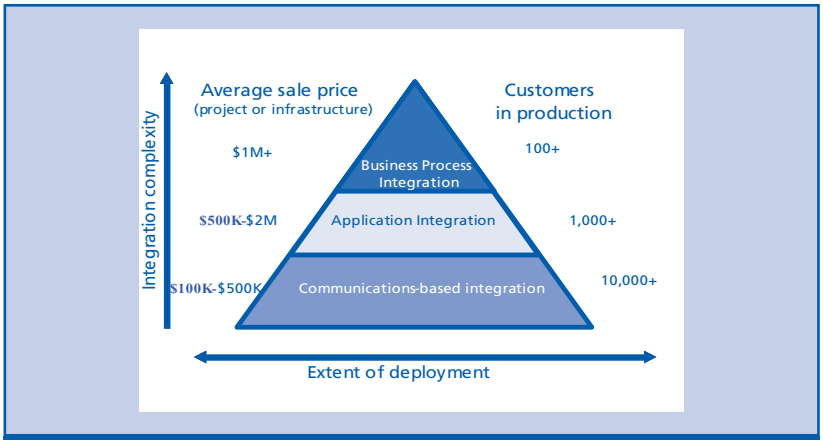


Figure 3.2: Simplified EAI market representation

Notice also, in Figure 3.1, that Development and Production tools span all layers. Because of the various market transitions in the past, these areas are sometimes only dealt with at the lower, more technical levels. In practice, insufficient attention has yet been paid to the need for equivalent development and production tools that are applicable at the highest layers.

Another point to recognize is that the layers have two vertical segments — Enterprise and B2B. Initially it was thought that external integration (B2B) was a different segment entirely from internal integration (Enterprise). However, as the economic realities of the early 21st Century set in, it became abundantly clear that the two were immutably tied together. Yes, there are still different elements of functionality that are required to serve each area. For example, security is usually much more key in an external integration project. There is more in common than had been presumed.

If Figure 3.1 summarizes the outcome of Phase 2 of the evolution of the EAI market, it leaves more than one conundrum:

- **should companies that are looking to implement integration solutions be analyzing all the solutions being offered across all these segments?**
- **should the support that a particular vendor offers for business processes be more important, as important or less important a consideration as support for file transfer?**
- **etc.**

The answer is that all of these segments may be of interest to an implementer in the long term. In the short to mid term, however, there are probably other more significant factors to consider.

Market adoption

As has already been discussed, the EAI market has been subjected to an enormous amount of marketing hype. For users it is important, but not easy, to cut through this to obtain a clear picture of where the market actually stands in 2003.

Today, most EAI vendors continue to push their value propositions up the stack, seeking to increase their leverage with the 'business' decision makers — and also to sustain their product prices. Indeed, a layman could be forgiven for thinking that the most important consideration in purchasing an integration solution is that it supports business process management.

The reality, however, is that it is the lower layers that are of far more importance to users and buyers. Furthermore, this is likely to remain true for the vast majority of EAI customers for some time. Figure 3.2 offers is a simplified representation of what the market has adopted today. It may seem puzzling that it is the lower layers which hold the attraction, especially when the upper ones are more easily recognizable to many business people.

Yet, there are several reasons for the delay in moving up the value stack, including:

- **the need for rapid returns on investment (rather than awaiting long-term gains)**
- **the difficulty of incorporating legacy systems**
- **the need for the scarce (and hence usually expensive) skills applicable to integration**
- **the lack of definition of many internal business processes (most businesses do not have an accurate idea or formal description of what their business processes look like)**
- **the organizational challenges that integration imposes.**

Most businesses nowadays operate in a climate of needing to show returns on investment (RoI) with a short payback time. Investing large sums of money on the basis that it will prove a wise investment at some time in the future is currently not acceptable. The fastest route to payback is usually the one that is most closely considered.

This obviously favors the simpler forms of integration because there is less customization and change required, a factor that is emphasized when you consider that few companies have the luxury of starting from a clean sheet. Legacy systems, software packages and integration technologies all have to be included in any integration solution.

Unfortunately, this fact is often forgotten by EAI vendors, since most of them are start-up companies with little past experience in legacy production operations. So connectivity to IBM mainframes, Tandem and DEC VAX systems often receive limited (or no) attention from vendors. Similarly, it is amazing, to me, how vendors overlook support for existing integration approaches — most obviously that for file transfer.

Other inhibitors

A second particular area of difficulty continues to lie in integrating legacy applications with application packages. Until a couple of years ago or so the EAI approach to this problem was for each EAI vendor to build its own specialized adapters or connectors (the terms are somewhat interchangeable although adapter usually implies a greater level of intelligent operations) for each application or environment. The problem was, and remains, that the integration vendors (quite logically) went for the major application vendors first — such as SAP, PeopleSoft, Oracle, Siebel, CICS and so on.

Figure 3.3: Communications-based EAI

Segment	Leaders/Challengers	Comments
Communications-based EAI - File - Message - Screen	IBM Sonic App Servers/JMSs Axway Mitem Jacada	IBM MQSeries is the de facto standard for messaging. For those wanting a JMS solution, Sonic leads the way although application servers are embedding better JMS servers all the time (for example, BEA and webLogic). Axway has perhaps the best EAI file transfer solution while Mitem and Jacada both offer good screen-based technologies.

While it is true that many customers have one, or maybe two, of these, it has also become clear that most have at least three or four other specialized legacy solutions that either do not have a wide market uptake or may have been written in-house. Even a relatively simple EAI implementation, which seeks to link two or three applications together, requires significant custom coding to develop the required connectivity to the 'uncommon' applications. It is no coincidence that, in most EAI implementations to date, the professional services component of the solution has been anywhere from 3-5 times the cost of the integration software.

A third inhibitor, to rapid progress up the EAI stack, remains the inherent complexity of EAI itself. It needs somewhat arcane skills which are almost certainly not all available in-house. As many synchronous developers have discovered to their cost, using an asynchronous form of integration uses a completely different design and programming paradigm than that used in the more traditional synchronous approach. Even assuming asynch. skills can be obtained, it then requires a special breed who can bridge between the business analysts and the IT specialists in order to implement a sensible business process management solution. The scarcity of operational tools for EAI, as mentioned earlier, places even more of a strain on the internal EAI skill set.

Climbing to the business process management level in the EAI value stack, another major issue has become apparent. Although executives are quick to see the potential benefits of being able to get a closer tie between their business process needs and the underlying IT implementation, it swiftly dawns on them that this is only of value if they can actually specify clearly and unambiguously what the current business processes are and which IT component implements what parts of the process today.

It turns out that there are very few companies who are even close to being able to do this. Often there are legacy components, for example, that execute important parts of the business but for which the documentation (and sometimes even the source code) has been lost. It can also be extremely difficult to describe the process actually being executed by any individual component in this situation. Even where there is a reasonable level of process specification, perhaps using such tools as Rational or Aris, few of the EAI vendors have the ability to exploit these specifications as a means to generate integration definitions

that can be used in their integration solutions.

The greatest inhibitor — the organization

Finally, maybe the greatest inhibitor to rapid deployment as well as development of higher level EAI solutions is the issue of departmental organization. While at the lowest levels of integration it may be possible to deal with an integration implementation from inside a specific IT department, as soon as the ties to the business process increase then business units want to be involved.

Segment	Leaders/Challengers	Comments
Application Integration EAI - Brokers - Adapters - Transformers	IBM TIBCO webMethods Sonic	IBM and TIBCO have maintained their positions in the top three since the market began, but webMethods has emerged as a strong contender in the third spot. Of the 'new wave' solutions, Sonic is gaining a lot of credibility with its standards-based solutions.

Figure 3.4: Application integration EAI

The difficulty is that existing competencies tend to focus on areas such as network communications, file transfer and perhaps messaging. Little, if anything, is known about the business process requirements that drive applications. Even less may be known about how bought-in application packages work or how these can be incorporated.

In summary, for EAI to deliver its full benefits it is necessary to be quite prescriptive about the technologies, models, interfaces and approaches that can be used. These management-based challenges often cross lines of internal demarcation. In turn this makes them extremely difficult and time-consuming to overcome.

New (and old) challengers

From the above discussion, the picture you obtain should be one of an EAI market where the EAI vendors are dragging customers kicking and screaming up their perceived value stack towards a promised land full of integration of everything, all the way down from the business processes that are fundamental to an organization's operation. Although the pace along this journey is not nearly fast enough for the EAI vendors' liking, most everyone seems to agree on the logic and the steps in the journey. Or do they?

The answer, of course, is 'No'. As often happens in business, when there is a market with large sales to be made and strong year-on-year growth, there are usurpers who are constantly trying to gain acceptance. The last couple of years have seen the evolution of the EAI market place twist and turn as a number of alternatives to the 'accepted' approach have gained credibility.

In increasing order of importance, three of these 'complications' are:

- **traditional work flow vendors entering the EAI space, from above the process level**
- **the revival of old technologies, in essence the application or pragmatism**
- **the arrival, or promise — depending on your viewpoint — of standards-based solutions which 'offer' dramatically lower prices.**

As soon as the EAI marketplace expanded to cover the concept of business process management — in contrast to the more technical requirement to provide interfaces between different application packages and solutions — a bridge was created by which work flow vendors might enter the picture. Vendors such as Staffware have built their reputations on being able to work at the business level with customers, helping them to organize their business processes and work practices in a more automated and efficient manner.

By comparison to traditional EAI vendors, these work flow vendors could reasonably claim to have a far better understanding of business process management and, in particular, all the complexities introduced when one or more business steps involve human interaction. By adding some basic EAI technology to their solutions, either developed internally or through partnerships, these companies are now starting to put pressure on the EAI market by threatening to disintermediate the traditional EAI vendors.

A second threat to the broader aspect of EAI activity is the resurgence of traditional technologies. The fact is that, although the hype surrounding EAI managed to persuade buyers that this type of technology was all new, it has actually been around for quite some time. With the weaker investment cycles forced on buyers (because of the general state of macro-economics and world trade) over the last few years, companies have re-examined their existing

investments in such technologies as file transfer and data integration — and found these to be cost effective and well understood.

One area where this has been particularly apparent is in the part of the EAI market dealing with business to business (B2B) integration. Initially the story went that, to achieve practical B2B integration in (say) the value chain, it was necessary to expose business processes and services to partners and have all members of the value chain interoperate in a closely-linked fashion.

This has, however, proved fraught with difficulties — not least about security, intrusion into a partner's systems and flexibility for further change to name but a few of the obstacles. On deeper examination it turns out that 'good old' file transfer can prove a perfectly acceptable solution to B2B integration needs in many cases, and it is much more simple to implement. Essentially, it:

- **provides a non-intrusive way to interface between two organizations, with clearly defined interface points for security purposes**
- **enables, and re-assures, the individual organizations that they can alter their own implementations behind the file interface without needing to involve third parties.**

Finally, the third challenge is that coming from the growing interest in standards as a way of forcing down prices and reducing the complexity of the overall integration task. With time this may yet turn out to provide the biggest shake up of all to the established EAI market. In the Java space, in particular, there have been extensive developments in standards for integration.

This has created a wave of nimble, sharp-teethed challengers to existing EAI market dominance. Simultaneously it is forcing traditional EAI vendors to embrace those standards. The overall downward pressure on price has already had an affect. Although it is still too early to be sure, it appears that standards will force significant changes to the EAI marketplace.

The standards effect

For example, a real alternative that has emerged to proprietary EAI offerings has come with the popularity of Java-based application servers. At the same time, standards are being defined in many different segments of the EAI space and seem to be gaining a strong foothold in the EAI marketplace.

Although the first development to affect the EAI space had little or nothing to do with Java — the emergence of XML as a self-defining way of formatting data — XML has quickly become a popular way of dealing with information flowing between different parts of different business processes. In turn this has considerably reduced the extent to which transformation engines are required — and where these are needed, there are specific varieties of XML for different needs, thereby making integration easier.

The JMS (Java Messaging Service) specification has also had a major influence on the EAI marketplace. Although the market already has a de facto standard for the messaging communications layer in IBM's MQSeries (renamed WebSphere MQ in 2002), JMS has gained a strong hold in the Java market. The evidence for this is overt. Almost all the alternatives — including MQSeries itself — have been obliged to offer some form of JMS connection.

Figure 3.5: Process Integration EAI

Segment	Leaders/Challengers	Comments
Process Integration EAI - Process modeling - Business rules - Process control	TIBCO Vitria IBM Staffware	Vitria had the first process-based EAI solution and still offers a good solution, but TIBCO has strengthened its solution and IBM's acquisitions of CrossWorlds, Holsofx and Rational should enable it to make a major challenge. Staffware is a highly credible challenger from outside the traditional EAI space.

The principle behind JMS, as with many of the other standards being introduced, is largely based on the 80/20 rule or something akin to it. Functionally, JMS implementations do not offer the range and depth of functionality offered (for instance) by a full MQSeries server. But JMS functionality is sufficient for many EAI implementations. Since JMS implementations are generally cheaper, are accustomed to Java and (theoretically) are vendor-independent, they are proving popular.

Today the JCA (Java Connector Architecture) specifications are in their infancy, and are incomplete. But the intention is to pro-

vide standards for interfacing to applications or environments. This is a highly complex area and it remains to be seen if the JCA specifications can be established in the EAI market. Nevertheless, if the specifications are completed and enough large application packages implement them, this will also remove a considerable amount of effort in the integration implementation cycle. Traditional EAI vendors will be exposed again.

Segment	Leaders/Challengers	Comments
EAI tools - Operational - Development	MQSoftware Candle TIBCO IBM	Of the specialist configuration/management tool providers, MQSoftware and Candle have the best EAI-appropriate solutions (IBM based). TIBCO has a strong toolset built in for its own product offerings. IBM's acquisition of CrossWorlds and Rational strengthen its EAI development tools portfolio considerably.

Figure 3.6: EAI tools

Finally, Web Services collectively covers a range of standard specifications that are aimed at easing the entire task of integrating applications and business processes together. The desire is to provide a standards-based way for business services to be published, discovered and accessed both within and across company boundaries.

Supporters of Web Services predict a future of globally available business process or sub-process services, accessed through a global direct and usable by any organization (or individual) anywhere. Fortunately for EAI vendors, the reality is a million miles away from this goal; it is unlikely that Web Services will have a significant impact on real-world EAI implementations for many years yet.

Current state of play

With all this movement over the years, the players in the EAI market have changed as well: early stars have faded, new challengers have emerged and more changes are likely in the future as the market continues to evolve.

Figures 3.3-3.5 summarize the current position, at least in my analysis.

Management conclusion

Improved integration has delivered, and is delivering, major benefits to many organizations:

- ***enabling business operations to be streamlined and automated***
- ***providing value-chain integration***
- ***delivering significant bottom-line benefits to businesses across the world.***

It has proved remarkably resilient, drawing investment attention even in the light of a dramatic economic downturn. In addition, EAI technology has evolved, going through a number of major transitions. It has still further to travel over the coming years. Possibly, however, the most fascinating aspect of the entire EAI story has been the ongoing struggle for market control and direction of market development between the various different types of software vendors, for example between:

- ***the pure-play EAI vendors, the new-wave standards-based vendors, the application server vendors and the major IT industry suppliers***
- ***those organizations that are actually implementing EAI solutions.***

The level of ongoing hype in the market has been incredible, whether from the vendors, analysts, IT press or service providers. But in the end, there are definite signs that the buyers are gaining control, at last. This market has not yet matured to the point where there is one leader.

The good news for businesses everywhere is that the final winner will probably end up being the vendor that learns to listen most closely to what customers actually need, and delivering this. Yet, it is a commentary in itself that something so obvious has yet to happen ...

Six things that could spoil the Web Services dream

Tom Welsh
Consultant

Management introduction

Web Services seem to be everywhere — in the IT press, at conferences and exhibitions, in analyst reports and above all in the proclamations of software vendors. Given the volume of pronouncements, it would be natural to believe that all practical obstacles have been overcome and that soon everyone's applications will be able to interoperate freely and easily, to the benefit of all.

Regrettably, this turns out not to be the case. While significant progress is certainly being made in some areas, there are a few awkward facts (and laws of nature) that stand in the way of universal deployment.

Before rushing to implement the new model, Tom Welsh considers a few of these potential 'project stoppers' under the following headings:

- ***hype and confusion***
- ***performance***
- ***integrity***
- ***security***
- ***ontologies***
- ***configuration management.***

Hype and confusion

Ask a dozen people to define 'Web Services', and you are likely to obtain several — probably 13+ — different answers. The most concise and conservative definition would be along the lines of:

> *'A Web Service is a software service that can be programmatically invoked over an Internet Protocol network, using (at least) SOAP, UDDI and WSDL.'*

That tells us what a Web Service is. But it gives little hint as to why:

- **we might want to use Web Services**
- **they (Web Services) should be considered an important advance in distributed systems technology.**

An alternative approach is to look at the range of software products that purport to implement Web Services. Microsoft was first off the mark, with its .NET launch in July 2000. Then came IBM, followed by Sun — after which most of the software industry piled in, afraid of being left behind. Soon every company that had been positioning itself as an Enterprise Application Integration (EAI) or e-commerce vendor had to have its own Web Services offering. In due course, most middleware and application server specialists followed suit, anxious to demonstrate 'added value solutions' as an alternative to the dreaded fate of being portrayed as 'commodity' vendors.

The further this view of Web Services is taken, the more complexity is added to what began as a rebellion against the complexity of existing middleware. This added complexity is anathema to many developers, who adopted SOAP and XML-RPC as the way for individuals, and small groups, to create distributed applications with an absolute minimum of overhead. Their invoices aim to contain half a dozen fields, all of them plain character strings: they want no part of the Towers of Babel that organizations like OASIS, RosettaNet and others are building under the guise of 'vertical industry schemas'.

It is easy to criticize mature, fully operational standards like CORBA and J2EE for being overly complex. Web Services looked as light as thistledown by comparison — when they were merely an exciting concept on a slide projector. But CORBA and J2EE began small, too. Then they grew inexorably, year by year, as practical experience indicated the need for changes and additions.

David Young, the chief evangelist at Lutris Technologies, has expressed what we might call the 'back to Nature' argument very clearly: *"the problem with CORBA was it was a little too big... [It was] boiling the oceans, being everything to everybody. SOAP is a much simpler notion of implementation independence. SOAP is absolutely key to creating a bold, beautiful, interoperable world."*

These words are becoming increasingly ironic, as IBM, Microsoft, OASIS, W3C, WS-I and others stack specification on specification, profile on profile, and vision on vision. In consequence I would argue that it is impossible to make any reliable statements about Web Services in general, until a consensus is reached about what Web Services really are (and are not).

Performance

Perhaps the first concern that occurs to anyone hearing about Web Services in detail for the first time is: 'won't that be very slow?' No one denies that HTTP is far slower than raw Sockets or CORBA's IIOP. Robert Orfali and Dan Harkey, in their book '*Client/Server Programming with Java and CORBA*' (second edition), cite the results of a simple benchmark that pings a remote server:

- **raw Sockets took 2.1 msecs**
- **RMI took 3.3 msecs**
- **CORBA took 3.5 msecs**
- **DCOM took 3.8 msecs**
- **HTTP took 827.9 msecs.**

The HTTP hit is barely half the story. Whereas conventional middleware transfers binary data in tightly-packed formats, SOAP sends human-readable XML documents across the Internet. Not only are the messages and replies much longer but they have to be marshalled, unmarshalled, parsed and interpreted.

In contrast COM, CORBA and RMI only have to do the marshalling and unmarshaling — and that is done automatically. When used in production, Web Services will often have to be encrypted, which will still further increase the transmission time and computing workload.

The critical question is whether Web Services will usually include a human being in the loop. If so, response time may be acceptable by Web standards. Otherwise — if we are talking about a pure machine to machine (or application to application) transaction — applications will take a severe performance hit from some Web Service invocations.

How much this will matter depends on quantitative factors such as:

- **how time-critical is this application?**
- **how many Web Service invocations occur for each program execution (or, if appropriate, per second of compute time); this question is complicated by the fact that a single Web Service might easily take several seconds to reply, swamping the relatively insignificant time required to process the data**
- **how long does each Web Service invocation take to complete: important parameters here include how many network nodes must be traversed to reach the server, how much latency is introduced by each node, how long the server takes to produce a reply and the likelihood of failures in the various nodes and network sectors involved**
- **does each Web Service invocation require round trips to a UDDI directory, a WSDL repository, and the target Web service itself; if so the chance of systematic or random delays increases significantly**
- **is the Web Service initially invoked no more than a concentrator or aggregator — which does its work by issuing further calls to other Web services; this is probably the worst case imaginable.**

It is an axiom of distributed systems design that the realities of bandwidth, latency and other forms of transmission overhead must be taken fully into account right from the outset. Thus, an object which is invoked many times per second should be located on the same computer as the software that is doing the invocation — even if this breaks encapsulation or otherwise clashes with general design principles.

Thus it may be better to relocate components bodily, rather than invoke them remotely through Web Services. But this is only feasible if the component in question either belongs to the owner of the invoking software, or is open source. Otherwise, licensing issues may arise.

Integrity

The distinguished software engineer Leslie Lamport once remarked that, *"[a] distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable"*. This risk can be mitigated only by painstaking planning and design — thereby potentially nullifying many of Web Services' supposed 'simplicity' advantages over CORBA and J2EE. If we tie all of our computers together in a huge network of Web Services, how often will large tracts of it be unavailable?

At the same time, simple Web Services have limited value. Many of the most interesting scenarios involve chains or cascades of Web Service invocations, with one service calling others to collect necessary information. Well and good, but quite apart from the performance implications, how will reliability be affected? If each Web Service is 99% reliable, then a chain of ten will be only 90% reliable. If each is 90% reliable, a chain of ten will be 34% reliable. And a chain of ten is relatively small ...

These figures may look pessimistic, but consider the precedents. Many existing Web sites are built as cheaply as possible, perhaps on a single PC server with limited bandwidth and no hot standby. They work adequately as long as nothing goes wrong.

Unfortunately, unexpected popularity counts as something going wrong, for it floods the server with a workload that the latter cannot sustain. The result: the server goes off the air (or is apparently inaccessible).

Conventional Web sites enjoy a built-in damping factor: the relative slowness of human beings, who must react to each page and then click a link to request the next. This will not necessarily be true of Web Services. As soon as a response is received, the client application will be ready to send another request. Services that commit the unpardonable sin of becoming too popular will likely be subjected to 'request storms' that may not be readily distinguishable from denial of service attacks.

Security

The Internet was never designed to carry the weight of responsibility that Web Services proposes to entrust to it. Rather it was intended to link a community of trusted and trusting individuals. Although a lot of thought went into making it as robust as possible, it was built to resist external stresses — not internal sabotage.

Today, the danger of sabotage looms ever larger. There are security experts who claim that they could bring down the global Internet within minutes, and — worse still — keep it down. In 1990, that would not have been so terrible. Even today, we could just about get along without the Internet. But if we are proposing to load more and more of our indispensable business communication onto it, and then follow up with business automation, it is vital to make sure that the network itself is hardened against attack.

Building secure, reliable Web Services is going to be one of the toughest challenges the IT industry has ever faced. It is natural to assume that those best equipped to meet this

challenge are the big software vendors with their huge reserves of experience and funding.

When we think about exactly who those vendors are, however, we are liable to get a nasty shock. The biggest of all is Microsoft, which does not exactly enjoy an unblemished reputation when it comes to security or integrity. Not very long ago, a major analyst firm recommended that enterprises should stop using Microsoft's Internet Information Server (IIS) altogether, on the grounds that it was too easy a target for hackers.

The simple fact is that today's Web is not a secure environment. Every day thousands of carelessly configured machines are broken into by humans or automated scripts. They can then be used as advanced bases from which concerted attacks can be launched on better protected targets. In addition, the current Internet protocols are not entirely successful in preventing 'spoofing' — a method of attack whereby one machine pretends to be another, thereby gaining undeserved trust.

The consequences of this situation are not so bad today, and the potential gains to attackers are perhaps not great enough to justify really determined attacks. But, in a world reliant on pervasive Web Services, incalculable damage could be done via spoofing.

On a more generic level, advocates of Web Services make some questionable assumptions about security. As one writer tellingly puts it: *"We will still need HTTP because that is a good way to sneak through the firewall"*.

This assertion has been repeated almost every time Web Services are discussed. The logic is:

- **protocols like IIOP cause difficulties with firewalls**
- **developers would prefer not to have to deal with such complications**
- **therefore it is best to use XML over HTTP, which slides through port 80 like a greased lightning (what the firewall administrator does not know will not trouble him, and everyone will be happy).**

Being able to pass invocations and data through a firewall at will is not an unmixed blessing. We spend good money on firewalls — tiresome though they are — so that we can gain some control over what communications enter and leave our domains. But SOAP ignores security, leaving system managers to deploy it over SSL or HTTPS, or to install such filters as they see fit within HTTP servers.

Ontologies

On hearing of Web Services, an experienced software engineer is likely to ask, 'what about metadata?' This leads inexorably to the discussion of ontologies — a topic that is both rather less esoteric than it sounds and much more important.

The word 'ontology' originally denoted the branch of metaphysics that deals with the nature of being. In computer science, it has come to mean a complete and consistent set of terms that adequately describe a problem domain. The need for ontology can be seen if we consider the difference between a conventional middleware system and one that relies on Web Services.

When an RPC call takes place, the form of the call and all its parameters are checked. If the call is to succeed, it must:

- **exactly specify the called function**
- **include the correct number of parameters, each of which must be of the appropriate type**
- **request the correct return type.**

This is all feasible, because the call and the called function are written by the same team of developers, as part of a single suite of software. Moreover, the two pieces of code are compiled together and can thus be checked for consistency by the compiler.

In the case of a Web Service, the call and the called function will normally be written by different developers in different organizations, often using different programming languages. There is also a more subtle disjunction: they will not be referring to the same metadata, or set of variables.

What happens if a call requests a piece of data named 'Order Status', and the result is 'Confirmed'? Does this mean that the order:

- **has been received?**
- **is in production?**
- **has been dispatched?**

What if the part number that is returned does not correspond to any of the known part numbers? Questions like these illustrate the potential pitfalls of converting a data source intended for human consumption to one destined to be 'interpreted' by a program.

There are dozens of co-operative efforts in progress, with the common aim of establishing agreed vocabularies for specific domains such as automobile parts, healthcare sup-

plies or banking transactions. So far, the main result has been a greatly increased awareness of how narrow and brittle these agreed vocabularies really are. Every additional party that comes to the table with fresh requirements further complicates matters. As for the need to accommodate change...

These considerations seem to negate the much-hyped vision of an explosion of creativity on the Web. Instead, progress will probably be channeled within the 'official' ontologies of dominant vendors such as:

- **Ariba**
- **HP**
- **IBM**
- **Microsoft**
- **consortia such as BizTalk, ebXML, OASIS and RosettaNet.**

The following quotation from Robin Cover's Cover Pages on '***XML and Semantic Transparency***' gets right to the point: *"As enchanting as it is to contemplate the apparent 'semantic' clarity, flexibility, and extensibility of XML vis-a-vis HTML (e.g., how wonderfully perspicuous XML <bookTitle> seems when compared to HTML <I>), we must reckon with the cold fact that XML does not of itself enable blind interchange or information reuse. XML may help humans predict what information might lie 'between the tags' in the case of <trunk></trunk>, but XML can only help. For an XML processor, <trunk> and <I> and <bookTitle> are all equally (and totally) meaningless. Yes, meaningless."*

Configuration management

One leading figure in the world of distributed software who takes a dim view of Web services is Sun's James Gosling — who is heavily identified with Java, which he was instrumental in creating. Speaking at the SIGS Conference for Java Development in October 2001, he described SOAP as an 'over-hyped rehash' of RPC that would evolve into a 'semi-chaotic' system.

Perhaps Gosling's most telling remark was that Web Services were *"really all about exposing the services that people have already been building, but exposing them to people who may perhaps use them in unplanned ways"*. This was an observation calculated to polarize opinion.

On one side are the people who fervently believe that such unplanned re-use could lead to an explosion of serendipitous creativity — a sort of Internet Cultural Revolution in which a million flowers would bloom. This is not necessarily a naive point of view, as it has been espoused by some undoubted experts.

Pierre Haren, CEO of iLOG, is one such. He runs a company specializing in software components for optimization, visualization and rule-based computing. *"This is about the whole world of computing moving to a more dynamic model,"* he said in an interview with ComputerWire's Software Futures. *"We are talking about software that can be updated dynamically, changed dynamically, and [that] works dynamically"*.

That may be a vision that fills some people with excitement. It fills others with pure dread, as they contemplate how the inevitable errors might cascade and spread unchecked through a world of highly cross-coupled software applications. We are already concerned at the rate of spread achieved by some computer viruses, most of which are disseminated through email. Although viruses are deliberately written and introduced to the network by human beings, there is no reason why harmful defects should not arise spontaneously in a world where software can be changed dynamically — eventually perhaps by itself.

Many who have seen the consequences of unchecked creativity in an IT environment fear that the sort of chaotic improvisation admired by Web services enthusiasts could lead to disaster if practised on a global scale. Evidently James Gosling is of this persuasion.

Related to the issue of shared meaning is the generally overlooked problem of maintaining consistency among scores or hundreds of Web Services, any of which may change from time to time. Microsoft, which has already grappled with these perplexities, adopted a crude but effective approach with COM: interface changes were simply forbidden. Instead, when a change was needed, it had to be published as a new interface. That way, users of the old interface could carry on unaffected — much as users can go on using older versions of software products, regardless of the latest releases.

Presumably configuration management will be taken care of in the shared registries and directories of the Web Services dispensation. As in the case of COM, some discipline may also be demanded of developers and users. But whose job will it be to enforce that discipline?

Management conclusion

Like space travel, Web Services technology is still in its early days. It would be foolish and misleading to confuse our existing communication satellite network with colonies on

the Moon and Mars, or with yet more futuristic possibilities like interstellar exploration. Yet something similar is going on every day when people, who should know better, blur the distinctions between:

- ***today's relatively simple, limited Web Services applications***
- ***medium-term futures like e-commerce***
- ***'blue sky' concepts like the W3C's Semantic Web.***

There are already many areas in which Web Services can supplement, or even replace, conventional middleware. Gradually, some niches are emerging where Web Services even enable ways of working that were previously ruled out. At the same time, work is going on behind the scenes to create and test-drive the specifications for tomorrow's more powerful, groundbreaking Web Services.

In the meantime, prospective adopters need to be keenly aware of the constraints within which today's Web Services operate. Issues like security, integrity, performance and configuration management need to be thought through at the stage of feasibility studies — not left for hapless developers to grapple with as over-optimistic deadlines slide by.

As for ontologies, the existence of a shared set of concepts and operations that can be unambiguously described in a common language is a fundamental requirement. Without this, any Web Services project should be deemed a non-starter.

Web Services management

Dr Keith Jones
IBM Worldwide Software Solutions

Management introduction

Service-oriented architectures for application systems have come to the fore in the last five years as an attractive way to boost software productivity through re-use of business logic as services. A massive cross-industry initiative to develop open standards supporting these architectures with a distinct focus on interoperability between platforms has now reached a point of critical mass. Many early adopters regard the move to service-oriented architectures as a key strategic initiative promising both cost-reduction and reduced time-to-market for competitive advantage.

As these standards mature, open source technologies, alpha and beta trial packages and initial middleware products are now available from vendors such as IBM, Microsoft, Sun and BEA — with function sufficient for early adopters. Already some enterprise Web Services are in production.

The reality of Web Services, particularly for those who are risk-averse, is that significant issues have been raised concerning access security, transport reliability, transactional integrity and management for Web Service applications. On all these issues there has been considerable progress in the last twelve months, with proposed open standards and early implementations. In this analysis, Keith Jones outlines the current thinking on the management of Web Services — which is possibly the last of the great obstacles to be overcome before their widespread adoption.

Focus on Web Service management

Few can have failed to notice the attention that emerging Web Services technologies have attracted in the last two years or so. Whilst visionaries have speculated that these technologies will revolutionize the software industry, others have quietly implemented their first production services, and yet more have experimented with ideas for developing innovative business value through deployment of new Web Services.

In the mid-1990s, researchers developed fundamental ideas that are now maturing as 'service-oriented architectures for building software systems'. At the turn of the decade the first technology prototypes emerged from leading software vendors such as IBM and Microsoft. Since then initial middleware product implementations have been made available by several other vendors.

At a very early stage it was realized that Web Service technologies would not be able to deliver their promised value without standards to shape the implementation of service solutions on the widest possible variety of deployed platforms. In the last two years a vast effort — involving hundreds of engineers from academia, software vendors and software users — has focused on the development of those standards with interoperability in mind.

That said, the focus is only now turning toward the management of Web Services. Why is this? The answer is that, until recently, other more pressing issues have dominated the engineers' agendas, such as:

- **security**
- **interoperability between platforms**
- **transport reliability**
- **transactional semantics for Web Services.**

In effect, these shortcomings in the Web Services technologies available today have held back all but the most risk-tolerant of adopting users.

The manager-agent-resource model

Management of computer systems and resources is a well-established discipline supported by large numbers of vendors and their products. Many of these products, such as the IBM Tivoli or HP OpenView families, already implement open standards (such as SNMP and WBEM) and support resources of many types on a variety of platforms. Since Web Services are implemented as sets of resources — albeit of a new type — it seems reasonable to hope that existing systems management technology could be re-used or at least adapted to manage services as they are deployed.

The manager-agent-resource architecture (Figure 5.1) is the most commonly implemented model. Computer resources of many different types at the lowest level in the architecture are monitored and controlled through agents that in turn are accessible to manager components. At the highest-level, managers may collaborate with other managers to provide large-scale management capabilities.

Agents, and the resources they represent to managers, are very often co-resident within the same execution environment — although this not required by the architecture. A wide variety of resources — from corporate databases to departmental business applications, to desktop configurations and cell-phone directories — can be managed using this hierarchical approach. Agents and their managers can also be co-resident within the same execution environment, although this a less commonly implemented pattern.

In object systems, agents present an interface to their managers with methods that provide access to information about the state of resources and usage metrics as well as access to commands that modify resource state. For example, a management component may:

- **monitor message queue resources, by observing capacity metrics and thresholds**
- **dynamically shut down a queue that is consistently throwing errors.**

The presentation of information in this model may take many different forms. Critical events may be presented as lightweight alert messages whereas periodic heartbeat reports may be presented as relatively large standardized data-streams. The Distributed Management Task Force (DMTF) Common Information Model is an example of a standard datastream flowing within a typical manager-agent-resource system.

Web Service resources

The familiar approach to building service applications (Figure 5.2) results in a number of new types of resource that must be managed. The service description — implemented as an XML document conforming to the Web Service Description Language's (WSDL) grammar — is perhaps the most important resource type. WSDL descriptions must be carefully controlled as they provide a critical means by which services may be distinguished by consumers.

The service registry component (in Figure 5.2) is often mis-

understood. Open standards have been developed for registries that may be used on a worldwide scale to catalog the businesses and the services they provide. Such UDDI registries already exist on the open Internet (and others may follow) which are managed independently by:

- **IBM**
- **Microsoft**
- **SAP**
- **NTT**
- **HP.**

However, independent industry consortia and government organizations can be expected to establish shared registries for specialized purposes. Indeed, many more private registries will be established over time within enterprises for local use. All these will need to be managed as a resource type that is essential to the Web Service infrastructures that they support.

In the simplest of scenarios, however, it may not be necessary to manage a separate repository for service descriptions at all. A local file system containing a managed set of service descriptions — for use by developers within a LAN scenario for example — might be all that is required.

Java Web Service resources

Service implementations may be hosted by providers on a wide variety of execution platforms and developed using several different programming languages and libraries. The most popular platform for enterprise Web Services is undoubtedly J2EE. The language libraries provided by many J2EE vendors now reflect the Java management extensions (JMX) included in the latest J2SE 1.4 specifications (J2SE has now been incorporated into J2EE for enterprise systems).

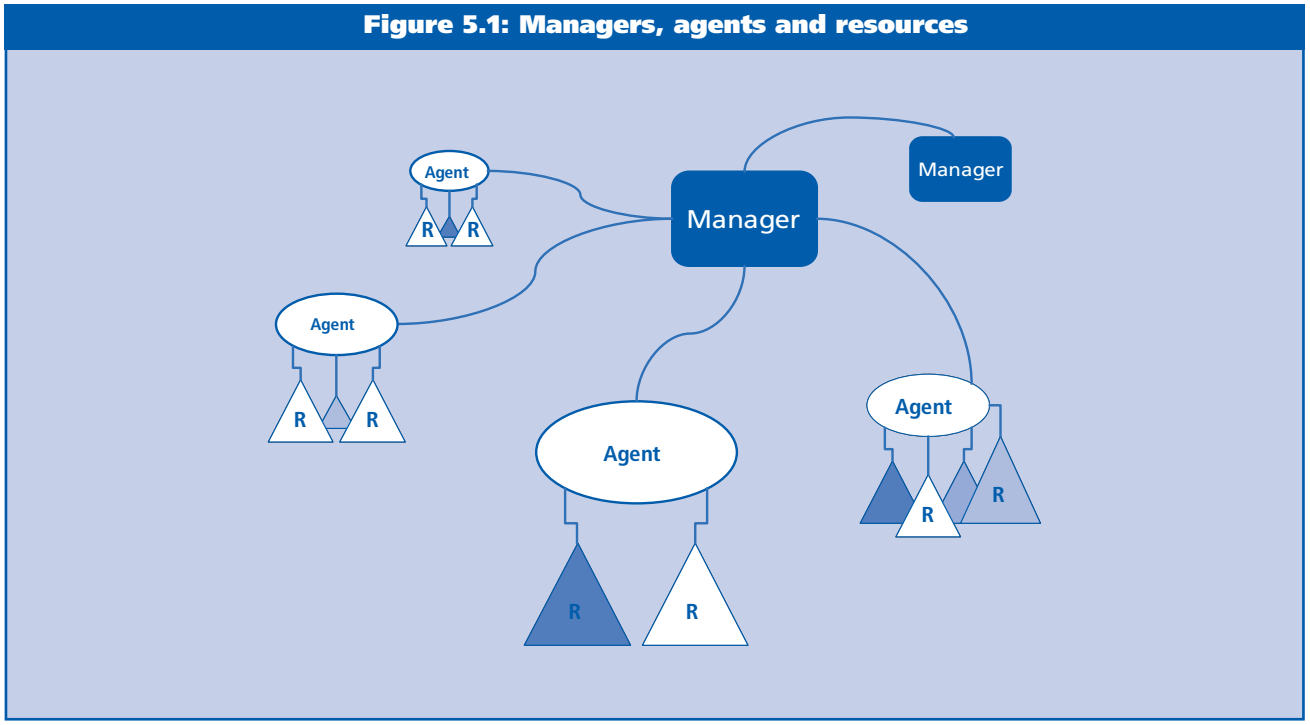
Web Services implemented on J2EE platforms, such as IBM's WebSphere Application Server, may be realized in several different component forms (JavaBeans, Servlets, EJBs, Java Connectors, etc. — see Figure 5.3) and deployed using a variety of alternative transport bindings. This allows for:

- **re-use of existing business systems where appropriate**
- **choice in configuring service systems to meet enterprise business goals.**

J2EE service implementations may be configured in separate containers for easy management and scalability or they may share execution platform resources. A given service interface could be implemented, and later re-implemented, using any of the alternative development patterns shown — without consumers being necessarily aware.

The same service interface could also be made available to

Figure 5.1: Managers, agents and resources



consumers with alternative binding protocols (this is not, however, shown in Figure 5.3). For example, IBM's WebSphere supports bindings for:

- **in-memory access**
- **SOAP over HTTP(S)**
- **SOAP over JMS**
- **RMI over IIOP using embedded Apache Web Services Invocation Framework technology.**

The service interface definitions, alternative bindings and service logic components are all managed objects in WebSphere deployment scenarios. J2EE 1.4 systems will standardize on this approach in 2003. At that time JMX will become a key element in J2EE management. (Other binding protocols may follow — and other application server vendors have similar lists of capabilities).

The JMX model

Open standard Java management extensions (JMX) provide an architecture and APIs for instrumenting Java application components and their execution environments for management. With protocol adapters to existing SNMP and WBEM standard managers, JMX systems will be integrated into large enterprise scenarios. As service consumers and providers are implemented using this technology, they too will be integrated as required.

Figure 5.4 illustrates the relationship between:

- **the generic manager-agent-resource architecture**
- **the JMX model**
- **probable J2EE deployment scenarios.**

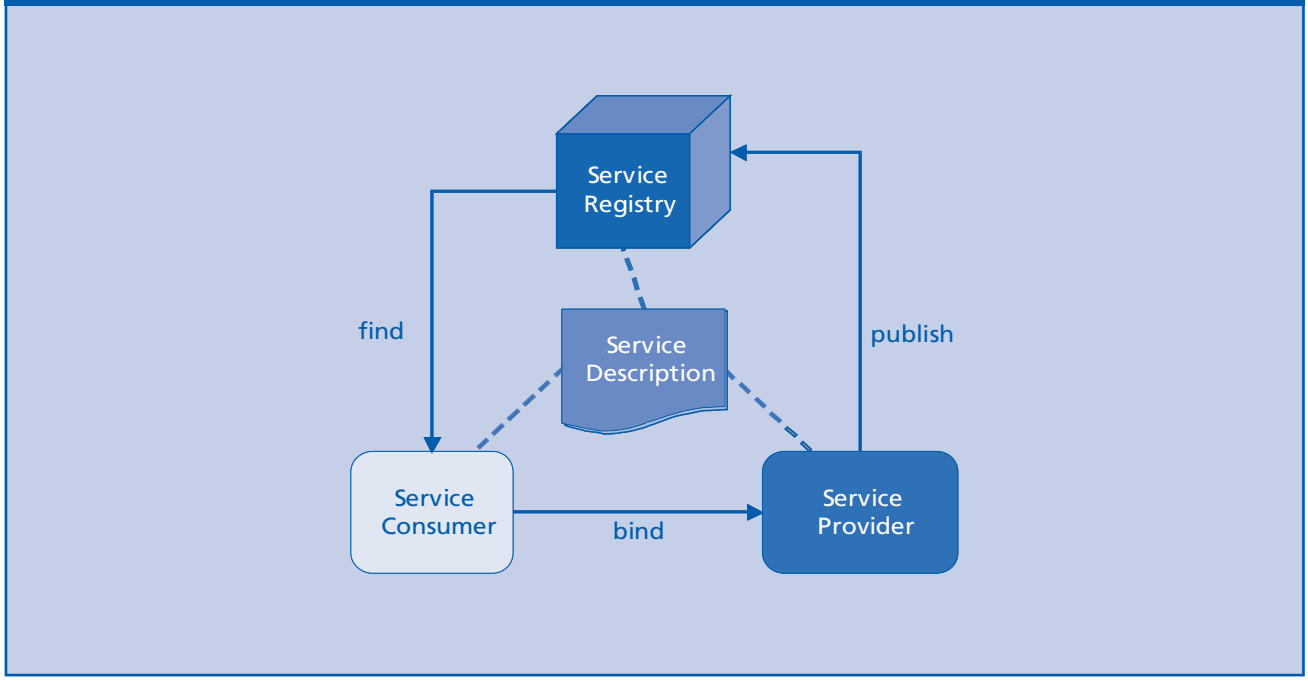
In the JMX model, management beans (MBeans) represent resources. MBean servers represent agents. Management applications communicate through standard interfaces to MBean servers to locate individual resources for monitoring and control.

In J2EE deployment scenarios both application and systems resources are represented by their corresponding management beans within execution containers. A management bean server retains knowledge of all the managed resources deployed within a container.

Whilst Mbeans and their resources are usually co-resident in a container, the Mbean server may be configured in a different container. Similarly the management application may execute in the same or separate container.

Manager applications may be distributed across a cluster of J2EE containers to provide comprehensive monitoring and control for larger sets of deployed resources. At the same time manager components may interact with external management programs using available protocol adaptors.

Figure 5.2: Services-oriented architecture



Over time they may also be accessible as management Web Services. A point to note here is that, in J2EE configurations, the container may take on the responsibility for automating the creation of the Mbeans needed for managed service resources — even though basic JMX allows for applications to create their own Mbeans and Mbean servers.

Managed Web Service interfaces

This discussion, so far, has focused on integration of Web Service resources with other existing managed resources in a relatively easy transition to Web Service-oriented systems. However, there is another dimension to Web Service management — the design of management function into Web Service interfaces and their implementations.

Since by definition Web Service interfaces are open to access from any standards-compliant platform (not only JMX J2EE platforms but also Microsoft .NET platforms, for example) it should be possible to incorporate management interfaces into WSDL service interface descriptions. As yet, there are no proposed standards for such additional interfaces. But, in this author's opinion, it would be reasonable to expect them in the near future.

A managed Web Service interface will include at least two different portTypes:

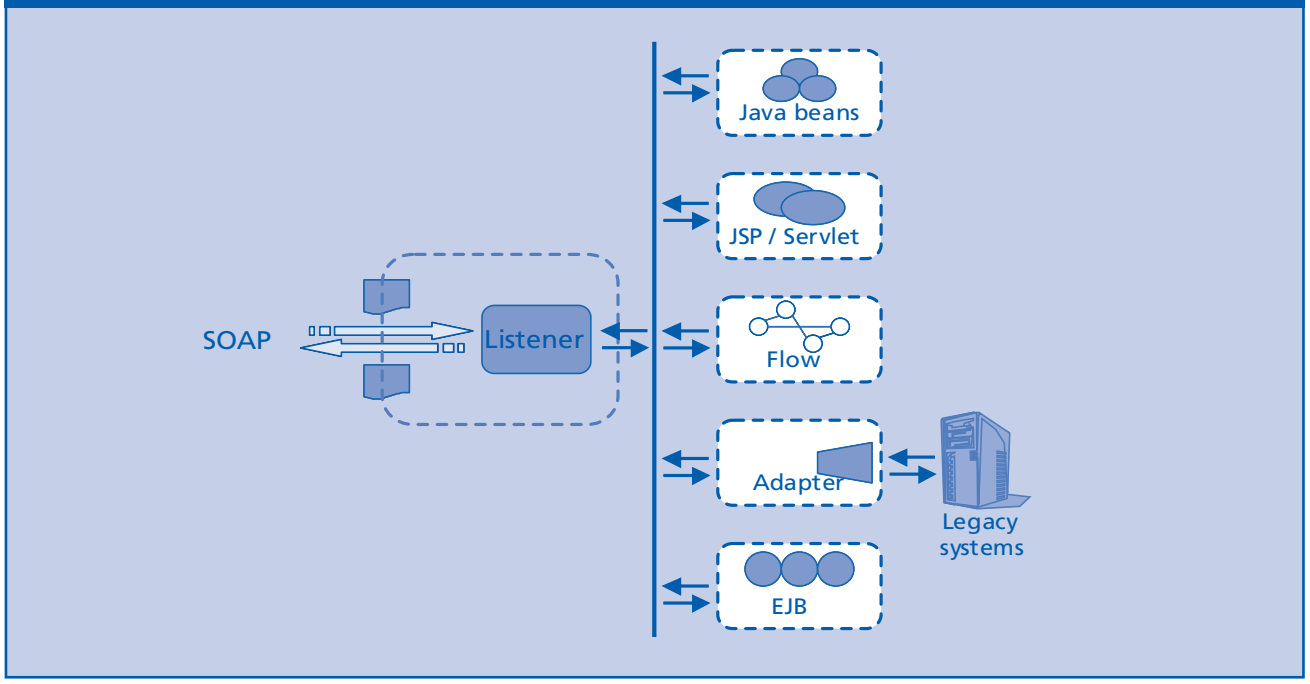
- **the first portType will define the functional operations that map service request messages to service response messages in the normal way**
- **the second will define management operations that provide access to usage metrics, configurations details and operational state for the functional service.**

As abstract WSDL portTypes are mapped to concrete ports at the time a service is deployed, it is possible to separate access to management interfaces from access to managed interfaces. This enables:

- **service consumer logic to avoid being complicated by the need to think about management issues**
- **appropriate security authentication and authorizations to be applied to each access stream in turn.**

Fortunately, it is relatively easy to see how operations on the service management interface can be mapped to implementations of management components within J2EE containers. After all, this could be the same mapping as is used for operations on the managed business service function. A recent development in the open source Apache AXIS project introduces a further useful technology for

Figure 5.3: J2EE service end points



handling management operations and functions (Figure 5.5).

The handling of SOAP service requests using a pipeline model introduces the possibility of intercepting requests and responses to apply common management policies such as logging, decryption/encryption, SLA metrics collection, performance monitoring and many more. It also introduces the possibility of automatically handling standard management request types without having to implement new service providers (indeed, this pipeline model is already included in the IBM WebSphere platform).

The link between managed Web Service interfaces in a WSDL description and the management infrastructure within an enterprise is critical to success. The advantage of having such additional operations included in a service description is that it allows for a service management infrastructure to be established across platform types using an approach consistent with the services being managed.

It also allows for management of contracts between organizations that are implemented using Web Services to be delegated to third parties (Figure 5.6) when appropriate. The several organizations involved in such a third party management scenario might be remote business partners. But it is more likely in the immediate future that they will be found in organizational divisions (within an enterprise) that individually own the business processes to be integrated.

Whatever the scenario, the management services monitor compliance to service level agreements and perhaps provide billing, auditing, non-repudiation and other valuable functions. These management services might also be a component of a larger set more generally associated with the concept of a service broker.

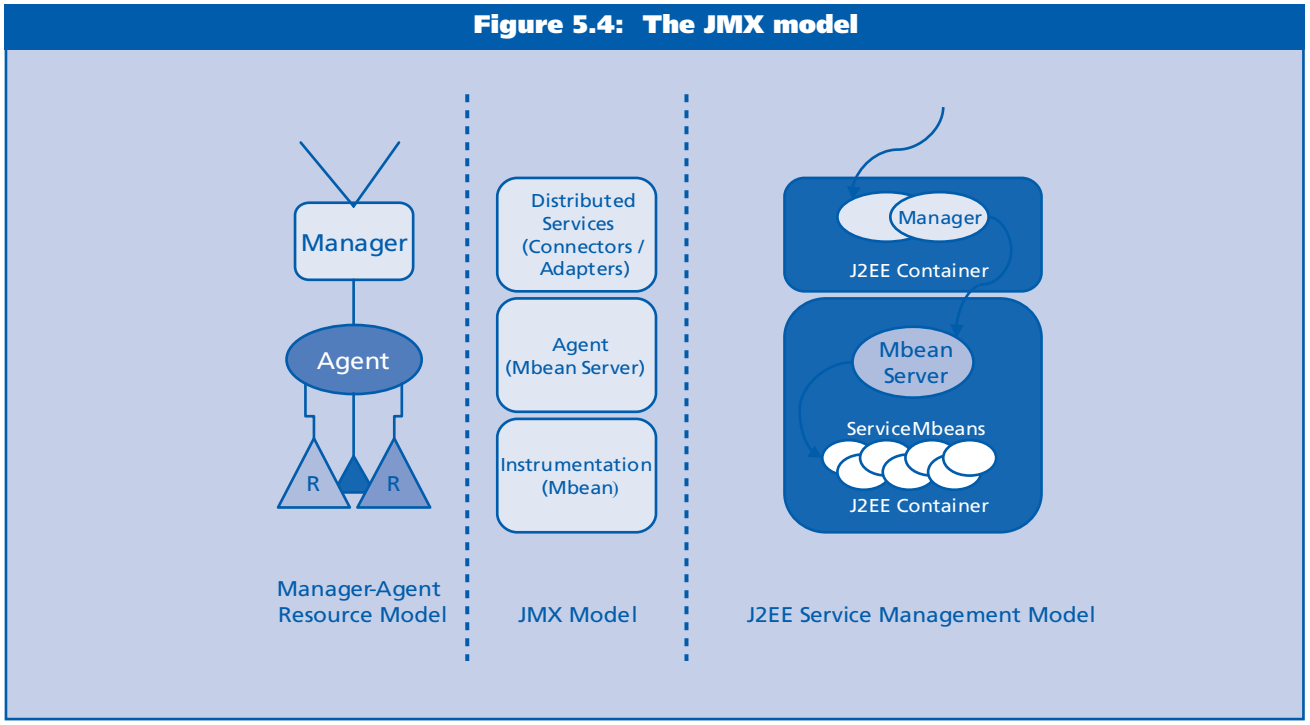
Managing to Service Level Agreements

Providing technical management infrastructure for monitoring and controlling services is probably the easiest of tasks required to establish a well-managed service solution. The more difficult tasks are those required to decide on the parameters for a service level agreement (SLA). But there are, as yet, no established or proposed SLA standards for Web Services.

A service level agreement is a formal contract between a service provider and consumers. In some scenarios this can be negotiated with some flexibility and in others the provider will dictate terms. For example, a consumer with considerable purchase power will usually be able to negotiate favorable discounts and priorities from amenable providers. On the other hand a government provider will dictate terms to consumers for services that are written in law.

The parameters most often proposed for Web Service level

Figure 5.4: The JMX model



agreements are those most easily measured using existing management technologies. For example, service availability may be quantified as a percentage up-time probability within published dates/times, and by mean-time-to-recovery times. Service performance may be quantified similarly as the mean-time-to-response or round-trip time with some additional parameters that quantify exception limits.

Further parameters to be found in service level agreements will reflect a compromise between consumer expectations and provider capabilities. Security, throughput volumes, versioning rules, remedies for failure and a wide range of other concerns must also be considered for service level agreements.

In future these parameters will become the basis for:

- **consumers wishing to locate services to meet their needs**
- **competition between service providers.**

Management conclusion

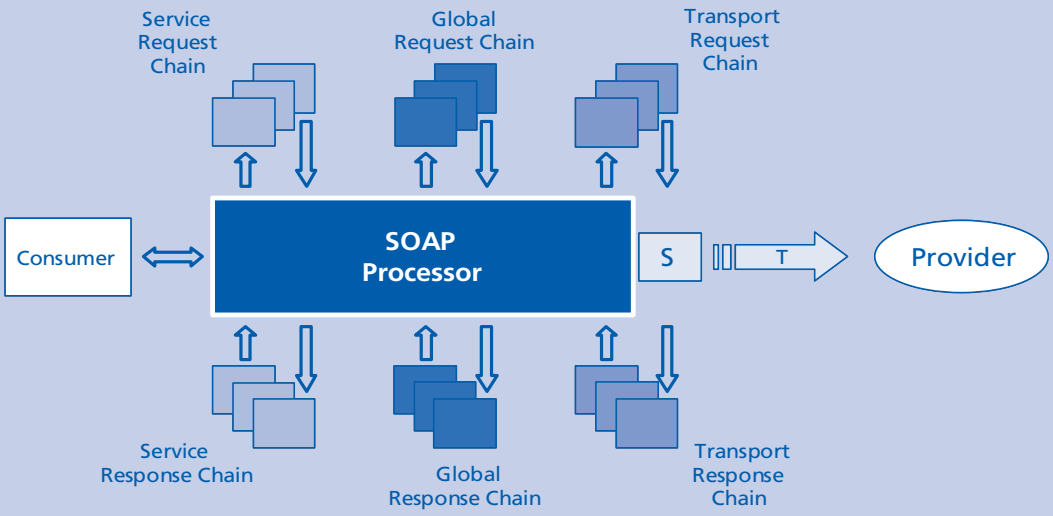
In spite of the hype surrounding Web Services, a very significant amount of effort across the industry is presently devoted to developing relevant open standards and compliance testing. All of this is focused on providing the foundation for distributed, platform-agnostic, service-oriented solutions that promise considerable business value to user organizations.

Many of the issues with Web Services that have caused potential users to pause have now been addressed in proposed standards and early implementations. For example, the roadmap for security — proposed by IBM and Microsoft in 2002 — identified building blocks for message integrity and confidentiality that have now been made available to early users. Similarly, support for transactional integrity and orchestration of service requests has been identified in proposed standards and demonstrated in 2002.

Management of Web Services is now coming to the fore as early adopters implement their first production service solutions. Users of J2EE platforms are well positioned. As the Java Community agrees on standards for integrating Java Web Services into J2EE 1.4 systems, it has also agreed on a JMX standard for management APIs to monitor and control Java application and system components — including Web Services.

The approach taken by J2EE vendors and users in the Java Community is consistent with standards such as SNMP and WBEM — which are already established for systems management. As new management products are made available by vendors, the expectation should be that Web

Figure 5.5: The AXIS pipeline



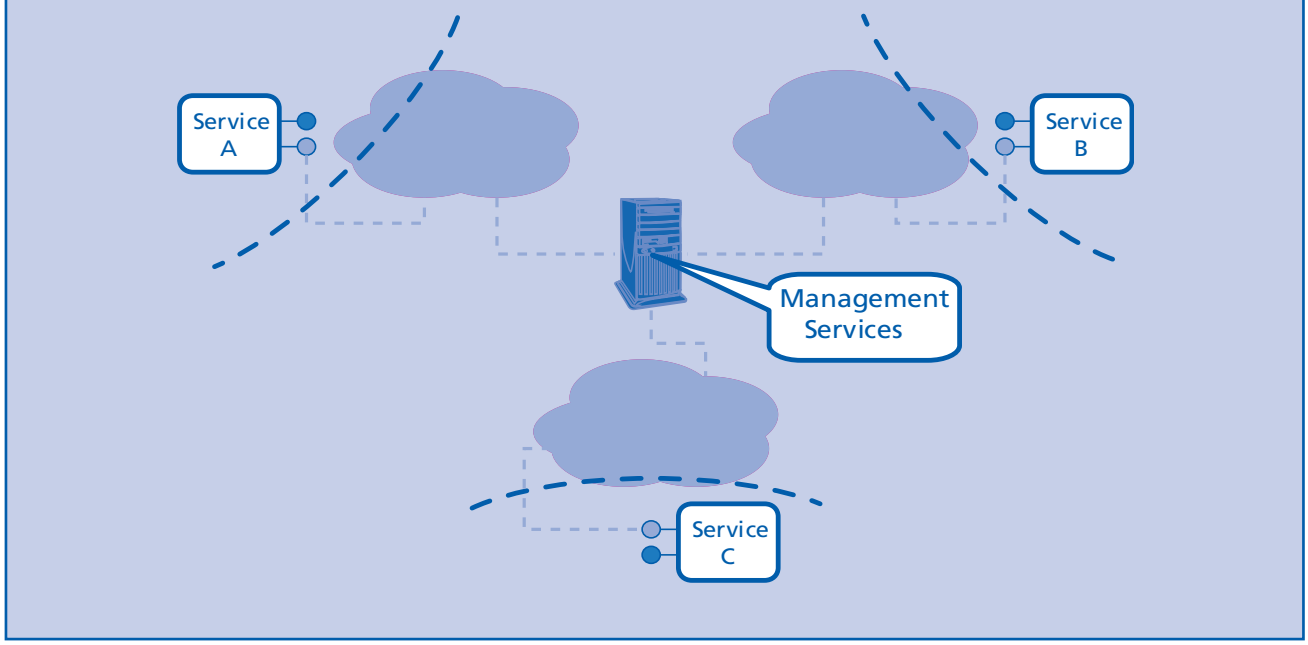
Services will be integrated seamlessly into existing enterprise management infrastructures.

However, technical infrastructure for management of Web Services is only one aspect of the whole. Business considerations must be expressed as service level agreements.

The parameters for these agreements have not yet been standardized. Service consumers will expect certain qualities of service to be guaranteed, whereas providers will at the same time want to maximize their competitiveness and limit their liabilities.

Agreement may be relatively easy within the constrained context of an enterprise network. But in a wider open marketplace for services this topic will need a significant amount of further work before Web Services management can be regarded as complete.

Figure 5.6: Third party management



Why do integration projects fail?

Roger Irish
Principal, Irish Beeston Associates

Management introduction

Application integration has not lost its attraction for business, whether applied internally or externally or both. While the capability and potential value of such business application integration remains high, experience shows that integration projects often fail, or fail to deliver anything like the expected or potential benefits.

This analysis — by Roger Irish, a practitioner in the integration field — considers how to assess whether integration projects have failed or are failing. It also examines some of the reasons for failure and this is supported by examples drawn from experience of integration projects from a selection of industries which:

- ***show how problems might have been avoided***
- ***illustrate what went wrong, and why.***

The subsequent analysis, on page 43, addresses these issues and proposes a project framework which will help mitigate these problems.

Setting the scene

Surveys of the spending on IT by a diverse range of businesses consistently highlight the integration of existing systems as a top priority. It is easy to see why — integrated applications promise value:

- **within an organization, through cost reduction and shorter process times**
- **outside, to customers and business partners through the ability to provide new capabilities such as self service and automated sharing of processes and information.**

Integrated applications and the associated integration middleware provide the foundation for initiatives like:

- **customer relationship management**
- **business to business integration**
- **business process management**
- **service oriented architectures.**

So, while integration is often focused on delivering tactical benefits — for example in the form of delivering legacy application information into new customer-facing Web applications or removing manual data entry steps in an internal process — it should also be seen as a catalyst for more significant changes. That said, however, delivery of the real value of integration software to the business — and the achievement of the maximum potential return on investment (and consequent shareholder value from the IT expenditure) — only comes when usage and objectives are aligned to business goals and vision.

What do we mean by integration project failure?

Integration project failure is used here to refer not only to those projects which are abandoned, canceled or never implemented in production, but also to those which fail to deliver on their stated objectives, or more significantly on their potential value to the business. Leaving aside for now those projects which, for whatever reason, are not completed, it becomes apparent that integration can be successful in the technical sense — where the integration software is installed, rolled out into operation with at least one integration scenario in production — but can still be considered a failure in the broader context (if it does not satisfy the business need). If an integration project is recognized as having failed, it is vital to examine the failure reasons in order to determine how these can be addressed. If this does not occur, the likelihood of any future value being generated from the sunk expenditure on integration software (and services) is significantly diminished.

Perhaps worse still are cases where failure is not recognized. This can occur, for example, if the objectives set for a project were set too low and thereby satisfied 'too easily'. In such instance, self congratulation may reign when remedial action is desirable in order that the larger (potential) strategic benefits and value of the project might be obtained.

It is, therefore, important to remember that most application integration software is generally sold on the basis that it is capable of delivering enterprise wide strategic value, and was priced accordingly. If the objectives of an integration project are purely tactical in scope then:

- **either the software considered for the purpose should have been appropriate for tactical goals (and with this should have come tactical pricing)**
- **or, perhaps more likely, the project team should have considered whether their objectives needed to address larger issues.**

Using this definition of integration project failure, it is possible to look at such projects and identify the signs that indicate failure or impending failure. Then we can consider the reasons for failure in more detail.

Identifying failed integration projects

One of the most common indicators of integration project failure can be seen in organizations that choose an integration middleware product for the enterprise, but fail to establish the product as an enterprise-wide solution. The consequence of this type of failure is that other business units within that organization evaluate and purchase other competitive integration and middleware products.

Although perhaps obvious, it is difficult to achieve strategic benefit from integration middleware if there are multiple competing products in use within an organization. Having to 'integrate the integration software' is frustrating, expensive and rarely worth the effort or investment.

Why do organizations purchase more than one strategic integration product? It is rare for the reason for this to be technical: in general the majority of products in the enterprise application integration software market are capable, albeit with differing degrees of ease, of integrating all applications that are capable of being integrated. Ignorance, about what others are doing and why, is one common explanation. Another is internal politics or positioning. A third comes with separate initiatives occurring nearly simultaneously.

One exception, to the desire for a single approach, occurs when integration has to happen business to business (B2B). If there is a requirement — or even a possible future requirement — for B2B integration then the software chosen must meet this requirement from the outset. In general integration middleware that can support B2B integration can be used within an enterprise. The reverse is not always true.

Not thinking strategically

In some instances an integration solution is implemented but neither the business nor key areas of IT are aware of it, or its potential uses. This problem tends be found in, though it is not limited to, large organizations. It is symptomatic of organizations that possess no central standards and no single authority with enterprise-wide responsibility for setting architectural or software guidelines.

In the worst cases separate divisions within an organization are defining their own requirements and evaluating their own potential solutions in complete isolation. This frequently leads to waste — in terms of:

- **unnecessary expenditure on software**
- **increased costs — in terms of operating, supporting and developing code for different integration software products**
- **the technical and financial issues that inevitably arise if two or more integration products themselves then have to be integrated.**

Put bluntly, there is rarely much business advantage in having two integration initiatives which are competing for strategic supremacy.

Building stovepipe solutions

A variation on not thinking strategically can occur when a bespoke integration solution is built using a technology (one example is enterprise Java) that was selected as meeting the requirements of a new application development. This happens particularly when 'stovepipe' solutions are commonplace. For the uninitiated, be warned: this can occur even though there is an integration infrastructure already in place that is applicable across 'stovepipes'.

While this approach will seem to possess some technical advantage — for the new application — it is likely that it will be at the expense of future flexibility, especially if there is a possibility that the data flow implemented will need to be shared with other applications in the future. Not using a strategic integration infrastructure to implement such a data flow creates two problems:

- **the existence of the flow may not be visible to potential consumers of the information, and so additional effort may be wasted discovering its existence**
- **when the flow is recognized, additional integration effort will likely be required to integrate this flow with the existing infrastructure.**

Over engineering against un-validated requirements

Occasionally the internal costs and charging mechanisms associated with using an implemented integration infrastructure are the cause of project failure. One interesting example comes from a large European financial services organization. It built a highly fault-tolerant integration solution based around several distributed messaging hubs, all designed with high availability in mind.

It was, and is, the practice of this organization to recoup IT costs through internal charge-back mechanisms. To achieve this it applied a per message charge which was associated with using the integration infrastructure. Unfortunately, with the exception of those applications which required such a high-level of resilience, these per message costs were prohibitively high for most business unit integration requirements. As a result, new projects that needed integration sought to buy alternative integration software to meet their needs.

Standing back from that organization, it was abundantly clear that the existing, implemented, software was more than capable of meeting any integration requirements that most business units had. Unfortunately it had become tarnished in the eyes of the business users, because it was associated with high charges. A second observation about this organization was that the high usage costs were the result of the way the integration software had been implemented. The culprit was not the integration software itself. Alternative implementations could have provided cost-effective integration for all business uses, and applications, that needed its capabilities.

Vision and goals

Lack of integration project vision most often arises where:

- **an integration solution is chosen to solve a specific application integration requirement**

- **there is a failure to recognise that the solution has a broader applicability across the business.**

One illustration of this comes from a large retail organization which chose an ERP package to replace some of its back office systems. It purchased an integration solution which possessed strong support for the ERP package already in use — in order to pass key business data into and out of the ERP system.

Project Dimension	Failure Characteristic
Scope	Tactical rather than strategic
Vision	Limited– addresses specific but not broad integration requirements
Objectives	Poor alignment with business direction
Tool Selection	Does not take into account existing software portfolio
Technical Design	Not appropriate for wide range of integration scenarios

Figure 6.1: Characteristics of failed integration projects

Subsequently a strategic integration requirement arose (or was recognized). Unfortunately, the originally adopted tool was not among the products considered for the larger requirement — because it was too specific and limited.

In retrospect, the original decision — to satisfy the ERP integration requirements by means of a tactical solution — could, with some planning and forethought, have also addressed the organization's integration needs of the future. In our experience, it is generally the case that, even if an integration requirement starts out being tactical in order for it to be successful, enterprise application integration should always have at least one assessment to ensure that the strategic dimension is neither omitted nor ignored.

Identifying failing integration projects

Recognizing that an integration project has failed to deliver on its potential value to a business is essential. Accepting this can be the start of a process which ultimately produces improved returns on investment, albeit further down the road.

Arguably though, of even greater importance is the identification of current projects which exhibit characteristics which suggest they may fail, or fail to deliver, either their promised or their potential value (Figure 6.1). Provided that remedial action is taken soon enough, it is our experience that it is possible to address all but the most fundamental of issues — thereby ensuring that the project is set back on track and capable of delivering the intended value to the business.

It is a software project axiom that the earlier a problem is detected and resolved the less costly it is to do so. This is especially true for integration projects. Some problems, such as ill-defined requirements, can be identified in the early stages of the project, and these present the best opportunity to make the delivery of maximum value to the business a core principle of the project. Even identifying problems during the development or delivery of a project still offers the opportunity to establish the core principles of the project. Once the project has gone into operation, it is considerably more difficult to rectify fundamental issues.

Once there has been some identification of the need for integration and the initial investigation stages are complete, it is often possible to discern symptoms of potential failure. Symptoms which can be identified in the initial investigation stage include:

- **overly lengthy product evaluations**
- **project scopes which are clearly tactical**
- **projects which are slow to start or to gain momentum.**

Perhaps one of the commonest problems observed is 'integration product evaluation paralysis'. This is characterized by an inordinately large period of time spent looking at different integration software products without selecting one for detailed evaluation or proof of concept trial or purchase.

In order for integration product evaluations to avoid becoming protracted (and this is all too easy given the difficulty of product comparison), it is essential that clear requirements are established before any selection process is started. All product vendors are capable of demonstrating unique features which their solutions offer. But, if these do not match your requirements, they should probably be considered irrelevant.

Assuming that the requirements are defined and agreed, and any criteria that will be applied to the software or ven-

dors are identified and stated in advance, then there is rarely a need for a paper-based evaluation to take more than 4-6 weeks. Some products may require the satisfaction of some form of 'proof of concept'. But, in our experience, many of the established products do not need that level of proof or confidence (which can often be achieved more swiftly and more usefully by talking with others who are already users).

Nevertheless, it can happen that integration software needs to be purchased to meet a tactical need, and even though enterprise application integration software is, by its nature, designed to meet strategic integration requirements. That said, in our view it is a mistake to accept that there is only a tactical problem to be solved without making an attempt to investigate whether any broader requirements exist.

All projects which result in the deployment of an application integration infrastructure solve a specific integration requirement, however strategic the overall objectives. Making the effort to identify whether there is a 'greater' strategic need early in the project lifecycle:

- **will help uncover the existence of additional requirements**
- **may also reveal that an alternative integration product has already been purchased.**

If a tactical approach is taken to each successive integration requirement, it is inevitable that early design decisions will not be appropriate for the projects that follow. In extreme cases it may be necessary to make substantial changes to what has already been achieved, in order to accommodate new integration requirements.

Building momentum

There are a variety of reasons why integration projects can be slower to start than traditional software development projects. Perversely, this seems to be more often the case for those integration projects which have strategic or enterprise objectives.

Where we have seen this happen in strategic projects, there appears to be a pattern that is followed. Because of the strategic goals of the project, and the relatively high purchase cost for the integration software, a great deal of effort is put into producing a solid business justification to gain executive approval for the purchase. So much effort is put into building this justification that, when the approval is finally obtained, there has been little or no consideration of what happens next.

The initial euphoria associated with a successful authorization is often followed by a period of indecision where the project team:

- **understands its destination**
- **has no idea how to get there, or even where to start the journey.**

Once the integration software has been purchased, and the project begins, there are other signs to be aware of, signs that may be indicators of different underlying problems. Questions to ask include:

- **are there clearly defined requirements for the project?**
- **does the project plan address all the requirements, objectives and critical success factors?**
- **is the scope well defined?**

Loss of direction

Analogous to projects that are slow to start are those that, once started, appear to be delivering little. Where this occurs it is generally not to do with resource shortages or poorly defined endpoints. More often it is a failure to appreciate the ways in which an integration project differs from a traditional software development project.

What often occurs is that the same approach is used for the integration project as for a traditional software development project. The result is a sequential delivery of function.

Project milestones are also often tied to the delivery of 'big things', such as a working infrastructure. This tends to result in longer project lifecycles. There is also an increased risk that what is delivered, is delivered too late for adequate testing. It is even possible that what is delivered no longer meets the requirements of the applications that are being integrated.

In organizations with established object oriented (OO) or component-based design and development working practices, we have seen this problem less often. There are key similarities between the approaches taken for OO and integration projects.

Inappropriate alignment

Another risk at this stage in the project lifecycle is that the infrastructure design becomes too strongly biased towards the needs of a single integration project. Typically this is the project undertaken in the first deployment of the integration software.

In our view it is vital, at this early stage in deployment, that all technical design decisions are assessed in the light of:

- **any additional application integration needs about which the team has become aware**
- **what the consequences of adopting each decision are, especially in terms of re-usability and flexibility.**

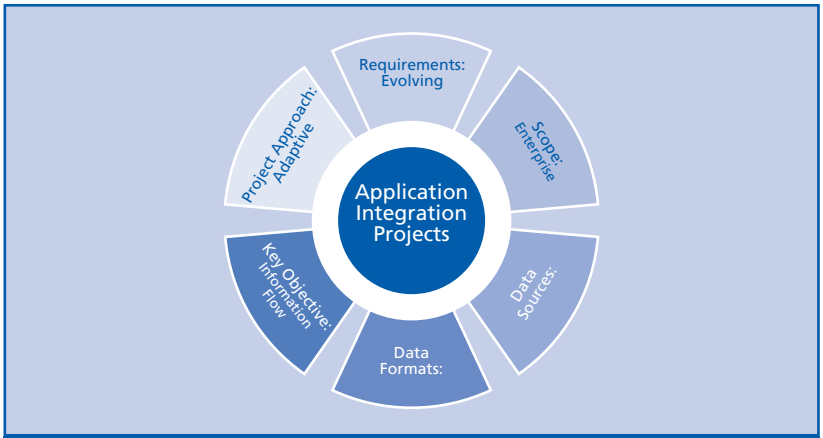


Figure 6.2: Integration projects are different

Underlying causes

Having considered the symptoms and problems which are indicators of failed or failing integration projects, the next step is to take a closer look at some of the reasons why these problems occur. In examining these it is important also to understand that there is not a one-to-one correspondence between the causes and the symptoms or problems. Problems often occur for a combination of these reasons.

Amongst the main causes of integration project failure are:

- **a failure to adopt a project structure that addresses unique aspects of integration projects**
- **a lack of alignment to business direction and vision**
- **poor communication**
- **unclear or undefined objectives.**

Project structure

Integration projects require a different approach to traditional software development projects because they are fundamentally different (Figures 6.2-6.3). For example, enterprise integration projects operate against a backdrop of continually evolving requirements, rather than a predefined set of documented and agreed requirements. In this context, the scope of integration projects tends to be the enterprise rather than the business unit or department.

These differences manifest themselves as project risks, and may impact upon project success if they are not considered and mitigated. They are of particular importance when deciding upon the integration project approach because the structure of an integration project must:

- **recognize these differences**
- **adopt practices that ensure they have no negative impacts.**

Business alignment

In order for an integration project to deliver maximum value, it must be aligned with what the business is trying to achieve. Projects which do not do this tend to deliver technology solutions which:

- **support a basic integration capability**
- **but are not easily extended to support the future needs of the business.**

For example, if there is a stated business direction to automate interactions with suppliers, then the product evaluation process must ensure that a business to business integration capability is a requirement. For businesses which operate as part of a larger group of companies, there may be a corporate goal to achieve a common approach for purchasing software. Where this is the case, an integration project may need to investigate whether there is a solution already in place elsewhere in the group. If there is, it may be faster as well as cheaper to adopt this.

Communications

In our experience, one of the most consistently overlooked causes of integration project failure is poor communications. This is not a simple issue which can be addressed by notifying assumed interested parties that 'the integration solution is live'.

Poor communications can be a cause of project failure for a number of reasons, including that:

- **IT staff are unaware how an integration solution might benefit them or their other projects**
- **the business units are unaware of how integration can provide them with benefits**

- **the business is unaware of how the integration solution might support alternative business models or ways of operating.**

Apart from not disseminating information about the benefits and value of the chosen integration solution, poor communications may have a negative impact on the perception of the solution when it is delivered. As is often the case when something new is introduced, there may be those (within IT or in business units) who have a negative opinion about integration software: they may consider it too expensive or perhaps they fear its automation is a threat. This may in turn lead to resistance to the introduction and use of this technology.

A brief example illustrates what can happen. A well-known UK retailer was undertaking a major strategic integration project which included a proof of concept that was designed to enable some of its developers to become involved with integration tools. Rather than be impressed with the benefits and ease of use of the graphical tools the software included, some of the developers involved formed a negative opinion of the software (and project) because it reduced the need for coding and programming skills. This was not addressed soon enough and negativism began to spread, with near disastrous project results.

Clearly a benefit to IT and the business may not necessarily be perceived as such to all those affected. That is understandable. But failing to take appropriate action to communicate can fatally undermine or weaken an integration initiative.

Project objectives

Whether the objectives set for a project are too easily achievable or impossible to deliver during the life of the project, the result is the same: the project will most likely fail. As has already been described, the first integration project undertaken usually has two broad objectives, to:

- **deliver a working integration infrastructure**
- **implement an integration solution for a specific business project.**

As project objectives go, these should be fairly easy to achieve. But, if this is all that can be demonstrated to have been achieved, it may be hard to convince others of the value of the solution. On the other hand, if the integration achieved has demonstrable financial value to the business, or the information that emerges can be shared by others, then the results will likely be seen to have value.

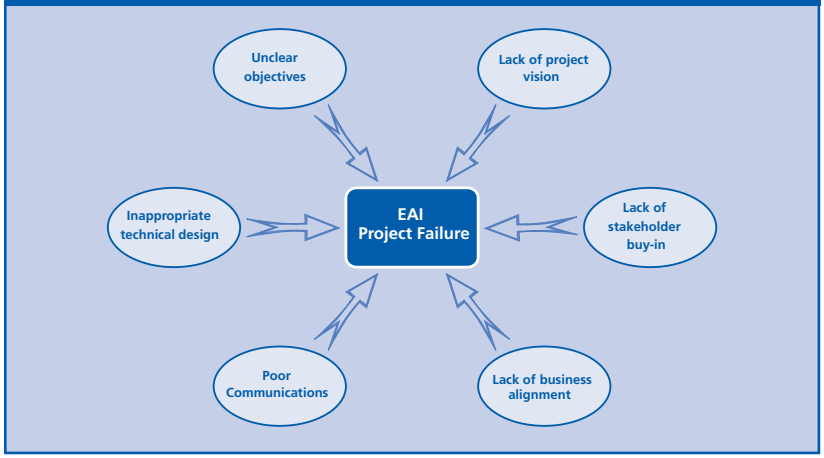
Similarly, if the objectives set are not achievable during the life of the project, however strategic their intention, then the project will also be seen as having failed.

Management conclusion

Integration projects fail, or can be deemed to have failed, for a number of reasons. Rarely is this because the technology did not work or was not capable of delivering what was required.

More often the failure is the result of 'softer' factors which relate to the project approach adopted. These include failure to take into account the ways in which integration projects are different to conventional application development, the need for greater alignment between IT and the business than is needed for single application development and why communication of the value must be widely and effectively broadcast.

Figure 6.3: Influences on integration project failure



Whilst IT departments generally excel at implementing and delivering technology, they are often not as good at addressing these softer issues. It is, therefore, imperative that integration software purchases deliver recognized value back into the business. Only then may the maximum potential value of integration be delivered.

Indeed, for integration to succeed it is likely that a framework for integration projects is needed. Using such a framework, it becomes possible to ensure that all areas relevant to integration are considered and then addressed (see opposite).

A framework for delivering successful application integration projects

Mike Beeston
Principal
Irish Beeston Associates

Management introduction

On page 36, Roger Irish examined why integration projects fail. In this analysis, Mike Beeston (a colleague of Mr. Irish) addresses what needs to be done if organizations are to deliver a successful application integration project.

His answer lies in the approach taken to such projects. He emphasizes the importance of ensuring that adequate efforts are made to take care of the considerations that lie beyond the technical challenges. The need for a fresh approach is explored. In addition, a framework for executing integration projects, that focuses on their unique needs, is presented.

The integration project challenge

In our experience, application integration projects too often are considered to be primarily about technology. The task in hand appears to be that of identifying the necessary technical activities required to implement application linkages. Such activities could include selecting suitable middleware products, adopting standards, developing code and so on.

While these activities are generally required — clearly application integration does require considerable technical effort — there is a further core theme to any integration endeavor. Application integration is also about implementing pervasive change in the provision of IT service for a business. Furthermore, that change should be expected to occur in many areas including, for example:

- **the introduction of new technologies**
- **the inclusion of new architectural capabilities**
- **the identification of new activities, roles and responsibilities**
- **the definition of modified organizational structures**
- **the changes of relationships between the business and IT communities**
- **the delivery of improvements to the overall capability of the business.**

Understanding, even accepting, that application integration is about change, and not solely technology, is essential if a coherent approach to what is typically a difficult project is to occur. In this context, an integration project must create a capacity for change throughout an organization.

As such integration is a change catalyst that must constantly realign, and be re-aligned, to evolving strategic needs. This is not trivial. Successful delivery will be heavily influenced by existing organizational structures, processes, capabilities, reward systems as well as culture. In essence, an integration project changes existing working practices. Its existence should indicate acceptance that the way applications have been deployed in the past is not adequate to support future needs. That said, it is inevitable that such change may be resisted — for it really is about breaking down established views of how systems are selected or built, deployed and used.

However, application integration projects should not be regarded as a manifesto for comprehensive change, although they may be part of such a program. Integration projects are concerned with the joint aims of delivering increased agility and operational efficiency to the business. They must be scoped accordingly.

As said above, this is not necessarily simple. The fact that technology continues to advance provides ever increasing opportunities for improvements in its use. At the same time a significant portion of any IT department has a 'natural' tendency that is inclined towards maintaining the status quo. On the plus side this contributes to consistency in systems provision, though the downside is the potential constraint on delivery of essential improvements.

For an integration project, one that is responsible for introducing wide-ranging change, an approach is required which addresses this disparity and enables change across the IT division.

Integration provides indirect benefits

What kind of problem are we solving? Not a business problem directly, but a problem that is associated with application systems. Certainly a problem may exist within a business if two systems are not able to share common data or processes effectively. In practical terms, this would likely be apparent either as an operational inefficiency (thus costing more than is acceptable) or as a barrier to business change (thereby inhibiting agility). The cost is the sum of:

- **the effort to implement change**
- **the lost opportunity(ies)**
- **any related competitive disadvantage.**

From a business perspective the problem is not that the two systems are poorly integrated. Rather, from a business person's perspective, it is that these systems appear to have been poorly implemented. Indeed, frustration about this has lead to many organizations looking to packaged applications or outsourcing as potential answers.

Indeed, one of the biggest challenges faced by any integration project is that the earliest beneficiary will be the IT department itself. The integration benefits experienced by business units will usually be indirect and only accrued over time.

Nevertheless, a typical cost benefit analysis needs to consider the tangible savings derived from more than efficient use of IT resources. The difficulty is that, although the promise of increased agility is an accepted driver for action, there are real difficulties in quantifying the secondary, business benefits that integration projects need to demonstrate to assure their return on investment.

Framework perspective

As described above, application integration projects are

about change. The framework presented here has specifically been developed to offer a process that can cope with evolution. The result is not only a technical capability to integrate applications. Of equal importance is the capability to contribute improvements in the provision of IT services to an organization's business units.

As such the framework seeks to deliver common sense. Indeed, experienced project managers may find many familiar reference points. Sadly though, many integration projects often suffer from a failure to apply these techniques and processes. At the very least the price paid comes with poorer returns on the investment made.

As important as the results is the fact that this framework terminology is not aligned to any formal methodology, project-wise or developmental. In fact it is quite the opposite. It is intended to be neutral and to map easily between established project management procedures and/or software development processes. Thus this framework provides a means to augment existing activities — without losing any of the value inherent in existing processes used in an organization. The framework is, therefore, not a substitute for formal project management; instead it offers the necessary integration-specific refinements to create a successful integration project.

Application integration projects are typically complex in structure. They touch many parts of the organization, are constructed from multi-faceted activities, and invariably are aligned to a business critical initiative. To balance this, the framework seeks to be as simple as possible and so improve opportunity for re-use across all aspects of the integration project.

Thus far, application integration has been described here as a distinct project. Arguably it is more accurately considered as a group of tightly coupled projects, perhaps better described as a program or, perhaps, a portfolio. Sample projects might include:

- **establishing an integration capability capable of addressing existing and future infrastructure and operational needs**
- **providing integration services to a specific combination of applications (possibly tightly aligned to a specific business initiative)**
- **promoting changes to application selection and delivery**
- **building a vision for integration.**

Such an approach is compatible with our framework concept. Indeed, we encourage tightly coupling the different dimensions. Breaking a large project into smaller, related projects helps in a number of respects, including:

- **finer definition of project goals**
- **earlier delivery of functionality**
- **improved controls, for example that applicable to project management and funding.**

The integration project framework

Our integration framework maintains three key elements throughout:

- **the iterative: it is widely recognized that a classic waterfall approach to project structure, though simple to understand, fails to deliver a sufficiently adaptive solution for integration projects; this integration project framework is implemented as a recurring cycle of repeated activities, preceded and concluded by necessary opening and review phases**
- **the incremental: each cycle is designed to deliver elements of the overall solution functionality; thus solutions are built incrementally because such an approach enables evolving requirements to be addressed within a project's lifetime.**
- **parallel: multiple activities running in parallel, when operating within clearly defined goals and towards a shared objective, are powerful.**

In the project lifecycle there are a small number of key phases representing a high level grouping of activities. At the heart are the two phases, of 'build' and 'live'. Each features a common set of five stages which are executed as a variable number of iterations, with their own discrete deliverables, that enable an incremental build. As every cycle runs, multiple work streams also run in parallel across the various cycle stages (Figure 7.1 shows the high level phases of the integration project framework).

Preliminary activities

The framework begins only after some preliminary milestones are completed. The framework focuses on:

- **the refinement of the integration strategy**
- **development of the solution.**

As such it is intended to extend, rather than replace, an organization's existing procedures. Similarly, the framework is not concerned with the selection of technology itself.

A number of tasks are expected to have been completed in advance of the initiation phase:

- **first, the application integration initiative strategic fit has to be established; the change that the project will generate calls for consistency with corporate strategy (for it would be a thankless task to embark on a project whose beneficiaries do not appreciate the value it may offer: worse, it may be perceived that a new strategy developed elsewhere in the business negates the requirement for the project benefits)**
- **second, the business case must have been accepted, including generation of a cost benefit analysis and funds allocation**
- **third, the key personnel, who will form the nucleus of the project team, must have been chosen.**

In the latter case, some of the significant roles include:

- **the Instigator — who will act as an evangelist for the project and will be the person responsible for identifying the opportunity for improvements through integration; instigators will likely have led the development of the strategic case for the integration project and will typically have responsibilities for IT strategy as well as presentation of the vision and benefits of integration**
- **the Sponsor will come from senior management and will be concerned with ensuring that the integration benefits are addressing real business needs; a sponsor can be regarded as the benefactor of a successful project — for the role must be active in terms of project control, perhaps chairing a project board**
- **the project Manager will maintain day-to-day control of the project, using standard organization procedures as refined by the integration project framework.**

In addition, there will be core team members. This may include external, as well as internal, team members who provide the key skills for implementing the specific solution.

Maintaining momentum

With multiple work streams running concurrently through incremental stages, and perhaps even crossing phase divisions, it is critical that there are no lulls spent waiting on stage completion prior to commencing new activities. The solution to this lies in good, traditional project management techniques. Good planning eases the flow of work.

What is important here is to focus on what is required to commence the next stage, rather than what represents completion of the current stage. From this perspective, it should be practical to start a new stage when sufficient work is completed in a preceding stage. At the same time, residual activities continue in parallel.

A simple example illustrates the effectiveness of this. Software installation tasks are clearly dependant on the availability of an environment on which to install. Ultimately this environment will require operational processes in place which support the service level targets, for example recovery procedures, availability management and so on. In this scenario, envision that a new server has been commissioned to contain part of the integration services. It is not necessary to wait for full operational support to be in place before beginning installation of the required integration software. Momentum is built and maintained by:

- **focusing on what must be delivered for the next stage to commence**
- **not giving undue bias to a tidy completion of all stage activities**
- **ensuring that it is a project management task to ensure that any loose ends are properly tied.**

Figure 7.1: Framework phases

Initiation phase

We are ready to begin. The project strategic fit is understood and funds are allocated. The core team is in place. The question now is, how do we ensure that the project gets the best possible start and the objectives set in the business case are met?

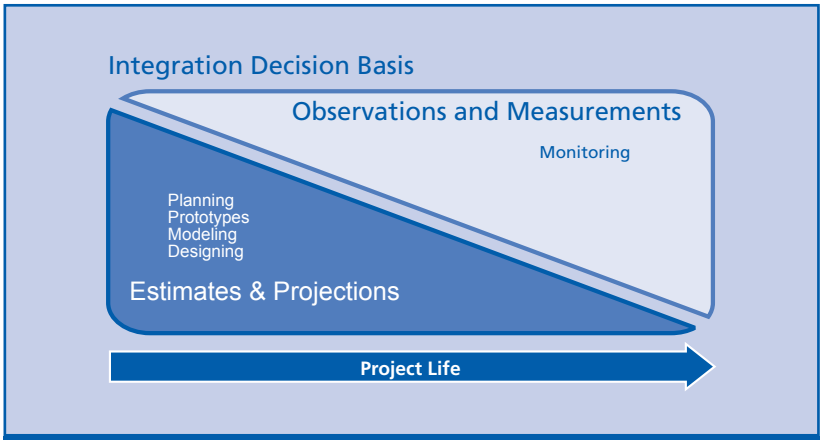


Figure 7.2: Integration decision basis

Until this point, the focus of the instigator and team has been to win project approval. Naturally some thought will have been given to how the project will be implemented, particularly when completing the CBA. Normally, though, the effort prior to this point is directed more to what, rather than how. For example:

- **what benefits will integration provide?**
- **what technology will be used in the solution?**
- **what must be implemented when the solution is fully deployed?**

Before commencing the project in full, attention must be given to how the project will operate and the integration solution will be delivered.

These may seem simple. But they will be fundamental to the success of the project. The work undertaken to win the business case is usually, in our experience, not sufficient to underpin the project development activities, though it will strongly contribute to these decisions. In the initiation phase, materials are developed — or refined — from the business case that provides the cornerstone for the project. In other words, initiation is an activity designed to establish overarching project qualities.

Furthermore, initiation is a team activity, perhaps the first since the project began and the team was assembled. Initiation is an excellent opportunity to ensure that all participants understand why the project will benefit the joint business and technology strategy of the organization. Working as a team, structured workshop sessions should be run that develop the following 'materials':

- **the vision: as the project is about change, so a clear vision of what that change really looks like is needed; this should be easily understood and capable of inspiring others, both in the project team and across the organization — for it is this vision which will help individuals understand not just what will be different, but why it will be better**
- **the purpose: if the vision offers a view of the ultimate project goal, the purpose identifies why the project team has been formed, explains why the project exists and communicates how it will contribute value to the organization; once the project completes, it is to be expected that an 'integration team' of some sort will continue to process new integration tasks — in this context the 'purpose statement' may be considered this group's raison d'être**
- **the key project management actions: this is the time to ensure that all project management activities are properly instigated, understood and documented (and this includes material such as project initiation documentation); essentially the project team must identify the outcome of the project — specified in terms of products, benefits, risks, assumptions and constraints — with this material forming the definition of the project and acting as a primary input to the next planning phase**
- **the approach: the project team must agree a common approach and methods for execution of the project which should include identification of shared ownership of objectives, collaboration techniques, relationships in and beyond the team, task management schemes**
- **the identification of stakeholders: the stakeholders, and their interests, need to be recognized for all stages of the project: the list of stakeholders may even expand (beyond those evident in the project roles) and could even extend outside the organization — for example to include regulatory bodies**
- **the scope definition: as integration is about change, the potential for change is likely to be substantial and it is essential to agree an**

effective scope definition that allows integration benefits to be created

- **principles: the team must establish principles that represent the fundamental way of working and building application integration solutions; these principles should be designed to support the project vision and purpose of the team; they need to be applied consistently by all team members and in cases where a principle may be too rigid, supporting guidelines should be agreed that refine the application of a given principle in a given situation.**

A common theme found in the initiation phase is that the products are generally static for the life of the project. Changes can be made to these project building blocks, although controls should be in place to ensure that any such action is performed with the endorsement of the project sponsor.

Primary planning sessions phase

With the groundwork laid in the initiation phase, the primary planning phase proceeds to create a general structure of the project. This is not a rigid plan but instead a 'roadmap' that provides a breakdown of major work streams running through the project and stage deliveries supporting the incremental build.

Planning itself is an iterative task. It is performed by team members with specific knowledge or skills that contribute to the development of discrete functionality. The project manager provides the point of consistency across all planning activities.

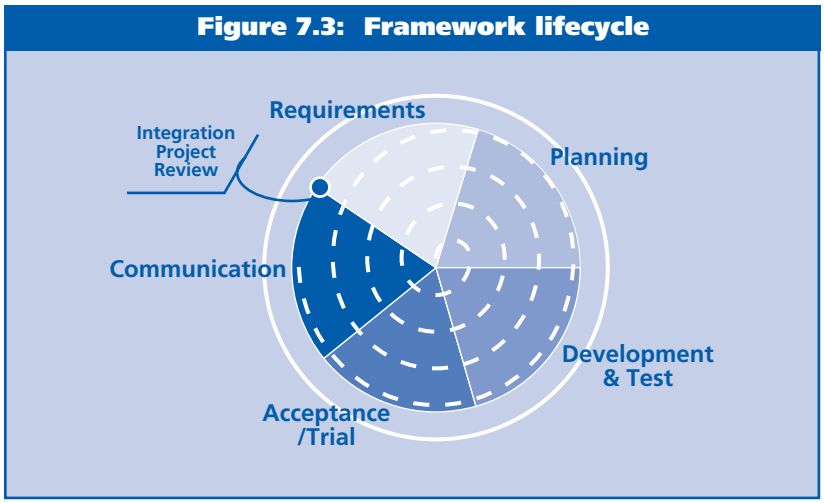
While project management is a constant stream of activity, planning is best executed as distinct tasks. In this primary planning phase, the team develop the overall shape of the project. Subsequent planning tasks are performed against more granular requirements with well defined goals. Some work streams are simpler than others, and their structure will sometimes seem self evident.

Initiation materials — such as the vision, scope and approach — have considerable influence on how the project structure will be developed. It is possible to begin the planning process without finalization of the initiation outputs, though the conclusion will be dependant on them.

To proceed, the output of these planning sessions must be validated against the initiation phase products and accepted by the team. In addition to the roadmap project plan, the following will also need identification:

- **improvement targets — for a common critical success factor is the demonstration of integration benefits; the team must identify how this will be measured in a manner that satisfies appropriate stakeholders (and this will often affect how subsequent project activities are performed to ensure the necessary data is captured)**
- **impact analysis — where the team must gather details of the existing environment in order to estimate the impact of proposed changes on existing business processes; projections need to include the current IT systems as well as elements such as organization structure**
- **work stream strategies — especially high level work streams, should be defined for the primary elements of the integration project; this will include developing strategies to handle change affecting people, such as new job roles, and for communications tasks, for example raising awareness of the integration vision.**

Figure 7.3: Framework lifecycle



In contrast to the initiation products — and perhaps counter-intuitively — the primary planning phase output is not a static project plan but a pragmatic structure against which subsequent activities are executed. Early planning is based on estimations and projections, contrasted against later stages that can be informed by subsequent measurement and observation (Figure 7.2).

Phases: build through live, in stages

The initiation and primary planning phases provide the structure within which the project will run. The next two phases are concerned with construction and use of the application integration solution.

The transition from early build activities through to subsequent production with a live system is achieved through multiple iterations of a common cycle of five stages. Multiple, parallel tracks of the project incrementally deliver components which combine to create the production system.

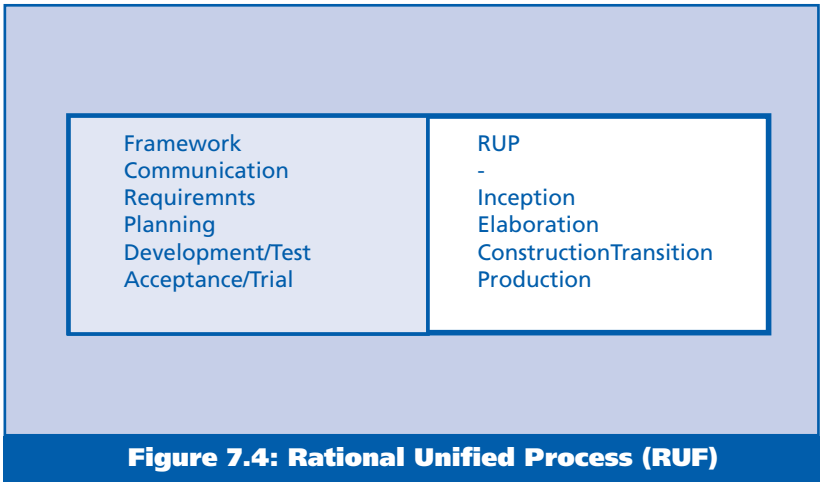


Figure 7.4: Rational Unified Process (RUF)

Once a live system is in place, the processes established during these phases will continue in new projects, as additional application integration requirements are addressed. The same techniques are applied to create the required work streams for these new tasks. Effectively, new projects are run to assimilate new integration activities into the established infrastructure. Figure 7.3 shows the framework's full lifecycle, with emphasis on the build through live recursive stages.

As noted earlier, the framework names are intended to be neutral of methodology terms. However, it is possible to map methodology names to the framework. For example, the Rational Unified Process (RUP) promotes a five stage project structure (Figure 7.4).

The mapping is not absolute. While the RUP does not contain a stage dedicated to communications, there is scope for such activities spread across other RUP stages. Conversely, the framework appears to provide a less granular approach to development tasks captured by the RUP in the construction and transition stages.

This mapping illustrates how the framework caters for application integration while allowing existing procedures to be exploited. Communication is critical to success for application integration, while the RUP is more concerned with project activities of development and testing.

The framework allows the team to approach integration development using construction and transition as 'sub-stages' in the framework. Although application integration is not a software (application) development project, parts of the project will require software development activities to be performed — and this framework allows known techniques to be applied.

The distinct needs of application integration are handled during these framework stages. Thus, the framework is:

- **a refinement to existing project management techniques**
- **a container for any existing development methodology**
- **a process that ensures application integration needs are properly handled.**

Communications

Effective communications are the lynchpin of a successful integration project. The most elegant technical solution will benefit no one if its value is not widely understood. Likewise, a solution that does not match the needs of the business will serve no purpose.

Without good communications between the integration team and its extended stakeholders there is a high risk of:

- **building a system no one wants**
- **producing a solution not aligned to the strategy**
- **loss of existing management support**
- **loss of budget and personnel.**

In the earlier build phase, stages are concerned primarily with:

- **awareness building**
- **objection handling**
- **training and skills development**
- **organization change**
- **progress reporting.**

As an integration project advances into the live phase, attention shifts to:

- **promoting application integration capabilities, including championing earlier project successes**
- **ensuring this solution remains the primary integration tool**
- **educating.**

A good communications strategy will contribute to better understanding between the various stakeholder constituencies, in turn improving the quality of execution of other cycle stages, especially requirements and acceptance. In this context, communications products delivered across the build and lives stages will address:

- **handling of negative responses to the change generated in the project**
- **personal understanding of how the integration solution will help the company to be more successful**
- **developing appreciation in the business of how increased agility is created by the solution, and how this might lead to new business opportunities and operational procedures**
- **establishing project working practices, for example, reporting procedures or use of email.**

Requirements

Identifying and correctly understanding requirements is manifestly vital for the success of the project. An essential activity in requirements gathering is the use of 'Process Definition Workshops' that bring together the stakeholders with the expertise in the solutions and processes being integrated. These facilitated sessions capture important information about how the existing systems operate, and contribute to creating improvements through integration.

Build phase efforts are characterized by a focus on:

- **what is required to create an integration capability; initially this will exhibit a bottom up bias along with identification of core, re-usable functions (for example error handling) and the information systems (hardware and software) needed**
- **identifying new teams and processes**
- **limiting the number of client project streams**
- **establishing process definition models.**

In comparison, the live phase will focus on:

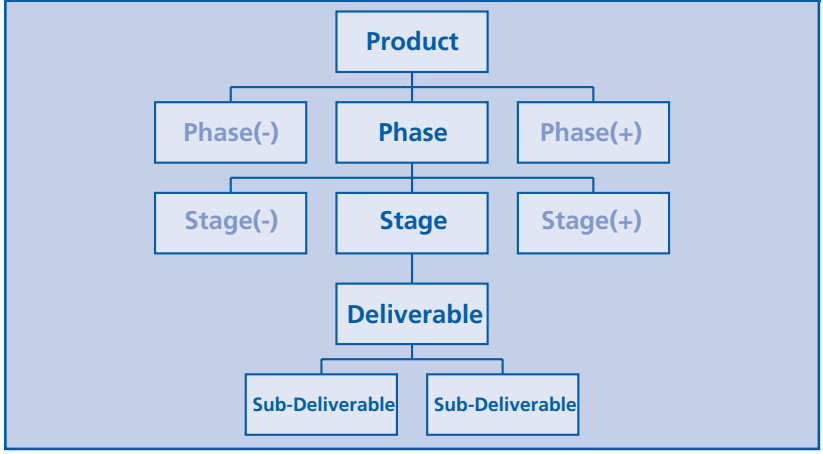
- **assimilating new integration requirements (new projects/initiatives and application portfolio changes)**
- **new core services**
- **new capabilities (especially those that are technology driven)**
- **any client project streams**
- **adopting a middle-out analysis approach.**

Planning

In the primary planning phase, work streams were identified and preliminary delivery cycles were established. In the iterative planning stages these plans are further developed:

- **improving estimations based on previous work cycles**
- **incorporating new and evolving requirements**
- **creating detailed task breakdowns and cycle activity lists.**

Figure 7.5: Planning and delivery relationships



A key contribution to the success of the integration project is the use of iterative planning techniques to ensure that complex integration tasks are divided into manageable activities and early project decisions adapt as knowledge accumulates.

Individual team member tasks are defined as part of a project product. For example, a product may be production ready messaging infrastructure, so a stage delivery would include full technical testing. The relationships between product, phase, stage and deliverable are illustrated in Figure 7.5.

Detailed task lists show how each team member will meet its responsibility for any given deliverable (or sub-deliverable). Along with the development of the project plan, the planning stage addresses more 'traditional' IT tasks. Analysis and high level designs are produced here for the planned deliverables, along with plans for testing.

Contrasting the build and live phases of planning, in build the following characteristics are most apparent:

- **the incorporation of multiple activities (requirements analysis, project control, stage/task reviews and stage initiation)**
- **identification of external dependencies**
- **evolution of standards and best practices.**

With a live system, planning includes the same range of tasks as the build phases but adjusts to handle:

- **new integration streams**
- **the integration team operating as a contained unit.**

Development and testing

Many integration projects activities in this, and the subsequent acceptance stage, are familiar ground. In the build stages, attention should be directed to:

- **design and coding activities**
- **early stages that are biased towards implementing infrastructure**
- **development of core reusable services**
- **prototyping**
- **establishment of testing techniques and harnesses.**

Once in the live stages it is a case of applying, and perhaps extending, established working practices to meet new integration demands.

Acceptance and trials

In this stage, work stream deliverables are accepted by the appropriate stakeholders. A deliverable may complete a stage or phase cycle, or may be an intermediate delivery, such as a working prototype to be shared and used to develop further requirements. Remember also that a stakeholder is not necessarily the ultimate beneficiary of the deliverable.

Through the build stages, attention should be directed to validation, especially that addressing quality acceptance and service level validation, etc. and confirmation of the overall integration system. As the systems move to production, the live stages appropriately shift focus to consider:

- **the additional tasks of capacity planning, impact analysis and change control**
- **how handover, to the operational team, will occur.**

Completion

When is the project complete? How many iterations of the framework cycle are required? These questions should be answered during the initiation and preliminary planning phases. The objectives for the project are agreed at that time, as well as measurements being identified to allow confirmation that they are met.

The number of iterations of the core framework will be derived from the overall complexity of the project. More complex projects should be broken into smaller projects and run as a co-ordinated programme. When completed, the legacy of the project is a pervasive approach to application integration that should be accepted throughout an organization.

Integration project review

At a predetermined time after the integration project completes, approximately three months later, a review should be performed. Key team members and stakeholders should regroup to perform a structured, facilitated workshop to validate that the intended benefits have been delivered (or the necessary environment to generate these exists).

The review workshop should also identify remaining obstacles to success. If appropriate, it can initiate or generate new activities — perhaps even new projects — to build on the success of the existing work.

Management conclusion

Application integration projects can deliver positive returns on investment as well as generate business process benefits. To do so, they must address complex challenges through the application of an approach such as that embodied in the framework described above.

In all integration projects, it is imperative not to underestimate the extent of the change being undertaken. Equally important, you must extend the project activities to include full communication to your whole organization about the activities and benefits.Otherwise, failure remains a real risk.

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]