MIDDLEWARESPECTRA

incorporating FINANCIAL MIDDLEWARESPECTRA

Contents

November 2003

Volume 17 Report 4

Enterprise integration moves onto new planes

Charles Brett
President C3B Consulting and
President, International Advisory Board
MIDDLEWARESPECTRA

Management introduction

In the executive mindset there is little doubt that the lack of integration within an enterprise (intra-enterprise integration) is a source of bewilderment and frustration. That the same should apply to integration between two or more different organizations (inter-enterprise integration) is more understandable — although no less acceptable to any one wishing to manage a 'well run ship'.

*This **MIDDLEWARESPECTRA** concentrates on the latest thinking about integration and how it can practically be achieved. Not only does it look at integration buses (or, as the Gartner Group prefers to think of them, as 'enterprise nervous systems') but it also examines the current status of two would-be alternatives:*

- **Grids that are capable of providing transaction processing**
- **a service bus with container middleware.**

Implementing a service-oriented architecture: a case study

This case study, from Dutch insurance company Agis Zorgverzekeringen, sets the scene for what can be done with integration today. It describes how an existing (Unisys) mainframe application — one which already worked well — has been extended so that its superior processing capabilities could be made available to more customers through more channels of communication.

The key message in this case study is that a service-based approach, if clearly thought through, can be applied to both legacy and new systems. A services based approach is not only applicable to the latest generation of software technology.

Enterprise Integration Buses

The second analysis looks at enterprise integration buses. The objective is to examine whether these have validity in an integrated enterprise.

The broad conclusion is that they do. Ancillary conclusions include the dismissal of the differences between a hub and and a bus approach (in an enterprise) and that, while you do need an enterprise-wide vision, not everything has to happen (be implemented) at once everywhere.

Raising EAI standards

When Enterprise Application Integration (EAI) first appeared in the mid-1990s, it was largely the province of a small number of vendors with highly priced products. These vendors competed but gradually were obliged to move up the value-chain, if only to sustain their chosen business models.

This analysis looks at the continuing impact of standards on EAI. It concludes that the effects are growing and buyers can expect to see greater and greater value from a broader range of vendors who are using standards as a way to bring superior products to market which inter-operate.

Enterprise integration: buses and brokers

This analysis takes a different viewpoint from the two preceding ones. Rather than finding integration buses or hubs to be impressive, it is — if anything — rather dismissive.

In part this may be due to the over-hyping of integration — which has undoubtedly occurred in the past decade. On the other hand it also confirms the importance of integration to organizations. Integration is a subject, and destination, that cannot be ignored.

Transactional Grids

In the first **MIDDLEWARESPECTRA** of 2003 the focus on integration, services (as in Web Services) and Grids was clear. This continues. Grids make progress. The increased number of organizations (commercial as well as academic) exploiting Grid technologies is evidence of this.

That said, as this analysis indicates, the path to commercial-type transaction processing Grids is proving neither easy nor fast. There is much to do before equivalence to what already exists in most business-oriented organizations is available on a Grid, never mind between Grids.

Service bus and container middleware

Finally, in this last **MIDDLEWARESPECTRA** of 2003, there is an examination of service bus and container middleware and what this may mean for the future. Middleware vendors are focusing on developing and implementing a new generation of infrastructure which is standards-based and focused on the service-oriented approach.

These new middleware products are emerging to link service containers in a network that provides reliable and scalable transport for standards-based service messages. Unlike existing messaging backbones these 'service bus' products aspire to offer simple any-to-any connectivity via services delivered either as extensions to MOM packages or smaller offerings based on a JMS provider with added support for XML messaging.

If realized, the value add will lie in the mediation achieved between clients and service providers and in the management of quality of service per defined agreements/policies. This is a formidable challenge, but is no less valid for being so.

Management conclusion

The net conclusion from all these analyses is that integration — and middleware — is flourishing and changing. It is flourishing because the need for integration is increasing, not diminishing. It is changing because of the impact of standards. At its heart remains middleware.

The new middleware might not be that envisioned two or five years ago. Nevertheless it is real, and making progress, albeit more slowly than either IT or executives might wish.

Implementing a service-orientated architecture: a case study

Peter Bye, Consultant,
Unisys Systems & Technology
Herman Schreuder,
Manager of Technology and Innovation,
Agis Zorgverzekeringen

Management introduction

Case studies are instructive for anyone considering a new project of a similar kind. They provide models for technical architecture and design, which can be valuable. They may also provide guidance on other aspects of running a project — what to do and what not to do, in other words. Such guidance can be helpful in setting realistic expectations about implementation effort and schedules, and reducing risk.

This case study describes a project undertaken by Agis Zorgverzekeringen (hereafter called Agis), a Dutch insurance company, in conjunction with Unisys and other partners. The result, called the Poseidon system, is used by Agis to process insurance claims. The system has many interesting characteristics, including:

- ***a clear business motivation, with associated metrics, lay behind the decision to develop Poseidon; it was not just 'technology for the sake of it'***
- ***it was not a 'green field' implementation: Agis was already using a Unisys ClearPath MCP mainframe application, called IKAZ, with a terminal-based user interface — and this system was stable, worked and is functionally rich***

- ***it was decided to build upon IKAZ (rather than start from scratch) in order to capitalize on the functionality available and minimize risk; as such Poseidon has evolved the old application into a service-orientated architecture, and built a new superstructure — comprising work flow management, process definition and presentation services — in a tiered architecture (in the Poseidon environment, IKAZ is no longer accessed directly from workstations or terminals but provides pure services, independent of any device or access channel)***
- ***industry-standard middleware was used to provide the integration infrastructure required as well as the application execution environment***
- ***high performance was a goal of the implementation; great care was taken in the architecture and design, and choice of products, to ensure that this objective was met.***

This case study briefly:

- ***examines the business background***
- ***describes the architecture and technical approach taken***
- ***lists the results achieved***
- ***concludes by discussing the lessons learned.***

Agis — the business context and process flow

Agis is a health insurance company focusing primarily on public health care — which accounts for about 90% of its business. It employs about 2,000 people and has around 1.8 million people insured.

The primary business processes are concerned with:

- **receiving claims for treatment directly from healthcare providers**
- **ensuring that subsequent payments into the providers' banks are made.**

Each claim has to be validated and corrected if necessary. This can include, in some more extreme cases, returning the claim to the provider.

In this market, service quality means speed and accuracy of claim processing. Agis's goal with Poseidon was to move the company into the leading position in a highly competitive market by applying sophisticated, automated work flow and process management combined with high performance and reliability. A goal of the implementation was to do this in a way which would introduce a great deal of flexibility to respond to future changes, both within the market or externally — for example if required by government or EU mandate. A final goal, as might be expected, was to optimize costs within Agis for each claim processed.

Consider what was demanded by the general process flow. The architecture and design of the system to implement this flow are described in subsequent sections.

Claims can arrive in a variety of forms, including paper, floppy disk, CD and X.400 mail. Increasingly, however, claims are delivered electronically, especially from the big healthcare providers (hospitals). Standards have been designed for this, using XML and SOAP. A new portal, called Vecozo, has been built to serve the healthcare industry. It provides secure connections between healthcare providers — some 40,000 in the Netherlands — and insurance companies, using the Internet.

The simplest case arises when a claim arrives electronically, is verified (that is, without requiring the intervention of any Agis people) and the payment executed; all this is automated, from start to finish. This can be achieved by:

- **applying rules for validation**
- **defining criteria for authorizing automatic payment, thus ensuring no human intervention.**

There are, however, a number of instances where human intervention is required. Leaving aside those cases where the claim arrives via a medium needing human processing (to enter it into the system), the other cases include those where:

- **there are errors or other exceptions in the claim, which then require some level of intervention to have these corrected; the corrections are made by Agis staff and the claim returned to the processing flow (only in extreme cases might a claim have to be returned to its originator)**
- **the claim is of a type or size requiring human approval — it cannot be processed automatically, even if it possesses no errors.**

Very small claims may also be passed through without human intervention. The rationale is simple. It can cost more to perform the human processing element than paying the claim. The criteria for deciding whether human intervention is required or not, for example by the value of the claim, needs to be variable and can be varied.

Logical architecture

Poseidon is built following a tiered, or layered, architecture, which is shown schematically in Figure 1.1. This illustrates a functional, or logical, model (its physical manifestation is discussed later) where the various layers are:

- **the top layer — presentation; this comprises the user workstations, or other access channels, which enter the central Poseidon environment through an appropriate user interface**
- **the next layer is the process layer which defines the various processes executed by the users as a set of services (the small circles labeled S1, S2 and so on); the definitions of the processes — and the services they use — are contained within this layer, as is a work flow management system**
- **the services defined in the process layer are executed by service components (the small circles labeled SC1, SC2 and so on) in the next layer, which is the service layer; a service may be executed by one or more service components, and a service component may be invoked by one or more services; the service components are not aware of the characteristics of the user devices but simply provide a function (or functions) to the services that invoke them**
- **the final layer is the data layer which is where persistent data (about customers, claims and so on) resides, in the databases.**

In this model, applications are constructed using a table-driven approach in the process layer. This defines a set of service components which implement the logic and database operations of the service.

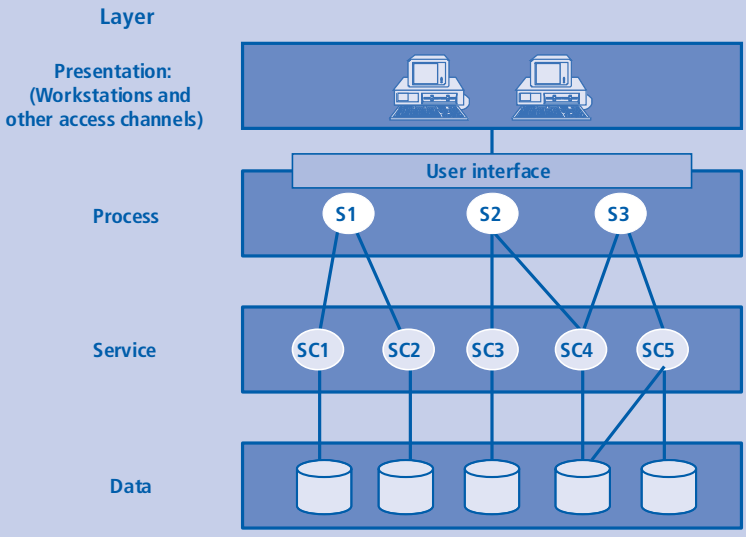
The service components themselves are self-contained and re-usable. In principle, they may be relocated among platforms if desired, without changing the services using them.

In addition, new service components, perhaps required by new services or extensions to existing services, can be implemented anywhere. The service invoking them from the process layer only needs to know:

- **the functions offered**
- **the interface, which uses industry-standard middleware.**

As remarked earlier, the specific design goals were to move away from the monolithic application structures of the past. Let us turn now to the implementation of the architecture.

Figure 1.1: Functional architecture



Implementation: an overview

Figure 1.2 shows the major elements used in the implementation. As noted earlier, before the implementation of Poseidon, Agis was already using IKAZ — running in a Unisys MCP operating system environment on a Unisys ClearPath mainframe.

IKAZ was implemented using a Unisys 4GL called Enterprise Application Environment (EAE — formerly called LINC). Although designed to run with terminals, and somewhat monolithic in its implementation, the application was considered functionally rich and stable. It was, therefore, decided to adapt the existing application into a service-oriented environment, to which EAE is well suited — because it can generate applications as services conforming to the Open Group DTP model standards.

Open Group DTP (formerly called X/Open DTP) defines a reference model, and associated APIs and protocols, for distributed transaction processing. The model supports the concept of a global transaction, executed over one or more systems, with 2-phase commit (2PC) options for database integrity.

XA is the defined standard for 2PC, and all major database products support it. The XA standard has also been incorporated into other distributed transaction models. Unisys and others have implemented the Open Group DTP model (the Unisys implementation is called Open DTP).

The ClearPath system forms the bottom two layers of the implementation, as shown in Figure 1.2. The connection with the process layer is via the Tuxedo ATMI interface (implemented in a way to be explained later).

The next layer, the process layer, is implemented in a UNIX environment. Applications are produced using the Uniface 4GL. In addition, Staffware is implemented in this layer, providing work flow management.

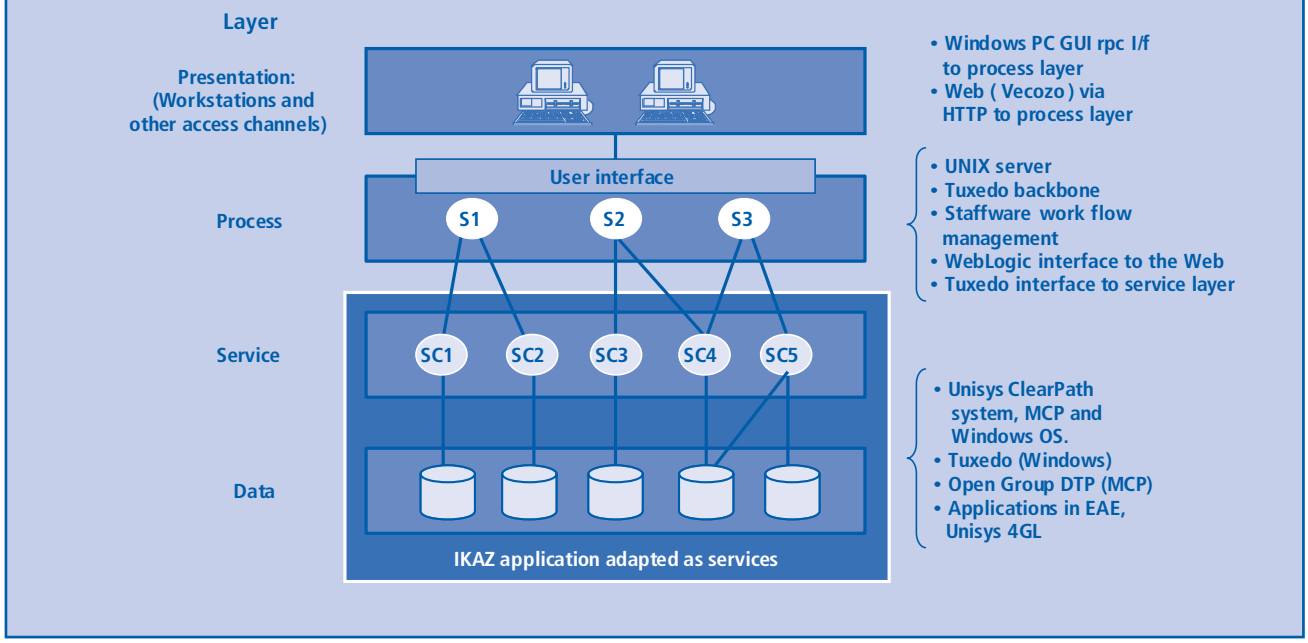
Tuxedo is used as a middleware backbone for connecting to the service layer. This was chosen because of its stability and performance, and has subsequently proved easy to extend into a J2EE environment. The layer uses an RDBMS (Oracle) to hold information.

The presentation (workstation or access channel) layer uses Windows PCs for the GUI workstations. The applications:

- **are generated using Uniface**
- **use a 'procedure call mechanism' to connect the workstations to the process layer.**

Web interfaces from the Vecozo portal (not shown in Figure 1.2) use a J2EE application server, running in the process layer, to provide the interface. BEA's WebLogic Enterprise is used, as it (BEA) provides both the WebLogic Server and Tuxedo as well as an interface between the two.

Figure 1.2: Implementation overview



Implementation details: service and data layer

Turning now to a more detailed description of the implementation, let us start with the service and data layers. The structure is as shown in Figure 1.3.

The service and data layers are implemented within the Unisys ClearPath system. These systems provide two operating system environments — with a close coupling between the two — in the same enclosure:

- **one runs the Unisys MCP**
- **the other runs Windows.**

Requests to the service layer from the process layer use the Tuxedo ATMI API, with Tuxedo FML buffers, which are self-defining — they are similar in concept to XML. These requests are received by Tuxedo running in the Windows environment within the ClearPath system. Tuxedo services then convert the FML buffers to View buffers, which are supported by the Open Group DTP model, which does not include FML.

The MCP environment in the ClearPath system runs Open DTP. The connection between Tuxedo and Open DTP uses BEA's eLink OSI-TP, which converts between Tuxedo and Open Group, using the OSI-TP protocols (as standard in the Open DTP model). This interface is used to invoke the service components, which are defined as Open Group DTP services, in the terminology of the Open Group model.

The logic and database access provided by these services are implemented as components within EAE. The EAE 4GL generates the code and the database, which is implemented with the standard DBMS (DMS II) within the MCP environment. Some interactions use 2PC between DMS II, which is XA compliant, and the database in the process layer. Services in the IKAZ environment can invoke services in other environments, for example in the process layer.

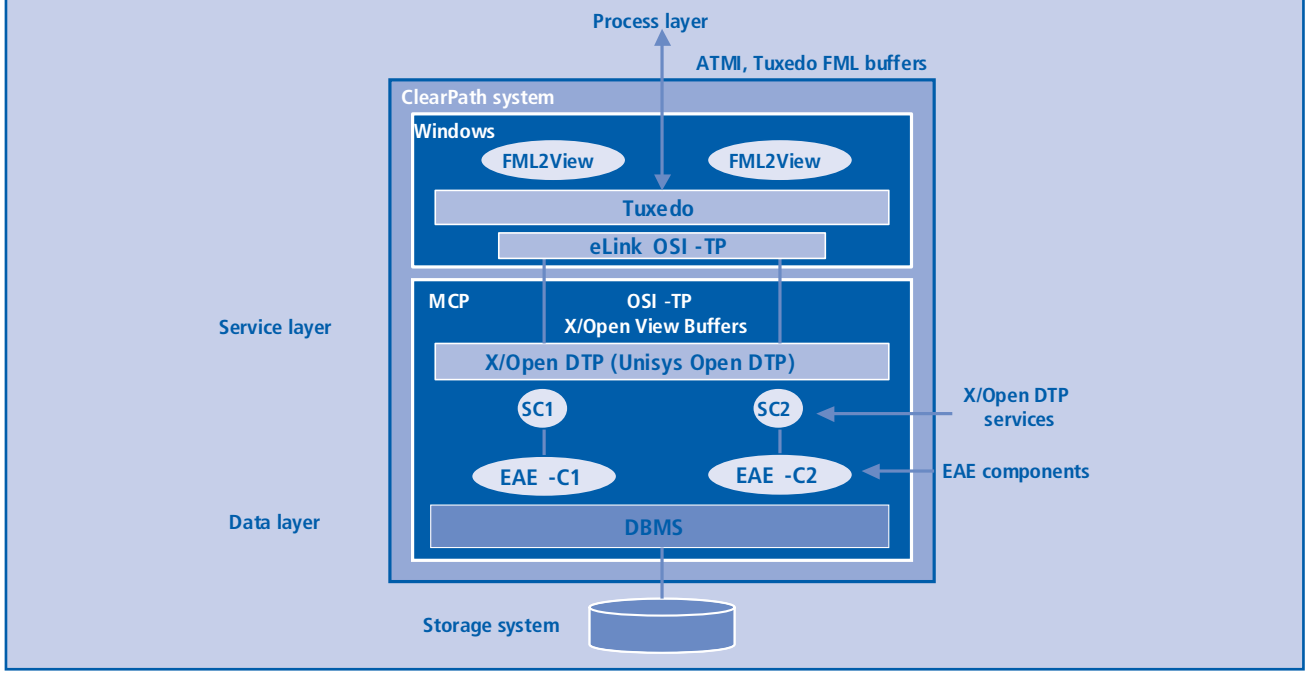
The EAE components were generated by adapting the IKAZ application. Although IKAZ was written using EAE, which supports the Open Group environment as a generation option, some reworking of the applications was required to produce well-defined, self-contained services.

Implementation details: process and GUI presentation layer

Now let us consider the process and presentation layer implementation. This is shown in Figure 1.4.

The process layer is where the application is defined at a high level, in the sense that it identifies the services to be executed in the service layer for each process. The tools used are:

Figure 1.3: Service and data layer structure



- **Staffware**
- **Uniface (from Compuware).**

Uniface is also used to generate the GUI used in the workstations. The middleware backbone again uses Tuxedo. Uniface is attractive because it supports a wide variety of middleware options. Processes are defined and stored in a process database, and are implemented as services — using Uniface.

These services identify the service components to be invoked in the service layer, where the logic is contained. The work flow procedures are defined in Staffware and identify the processes to be used for the procedures.

The flow through the environment (Figure 1.4) is:

- **a user retrieves a specific case from the Staffware work flow, selecting it from a menu of cases**
- **the processes — and hence the service or services (S1, S2 and so on in Figure 1.4), required to execute the flow — are identified**
- **the services use Tuxedo to invoke the appropriate service components in the service layer, which in turn access the database (as illustrated in Figure 1.3)**
- **the result is returned to the service invoking the service layer and then displayed on the screen for the user.**

This structure provides much flexibility. New processes can be easily described, using the service components implemented in the service layer.

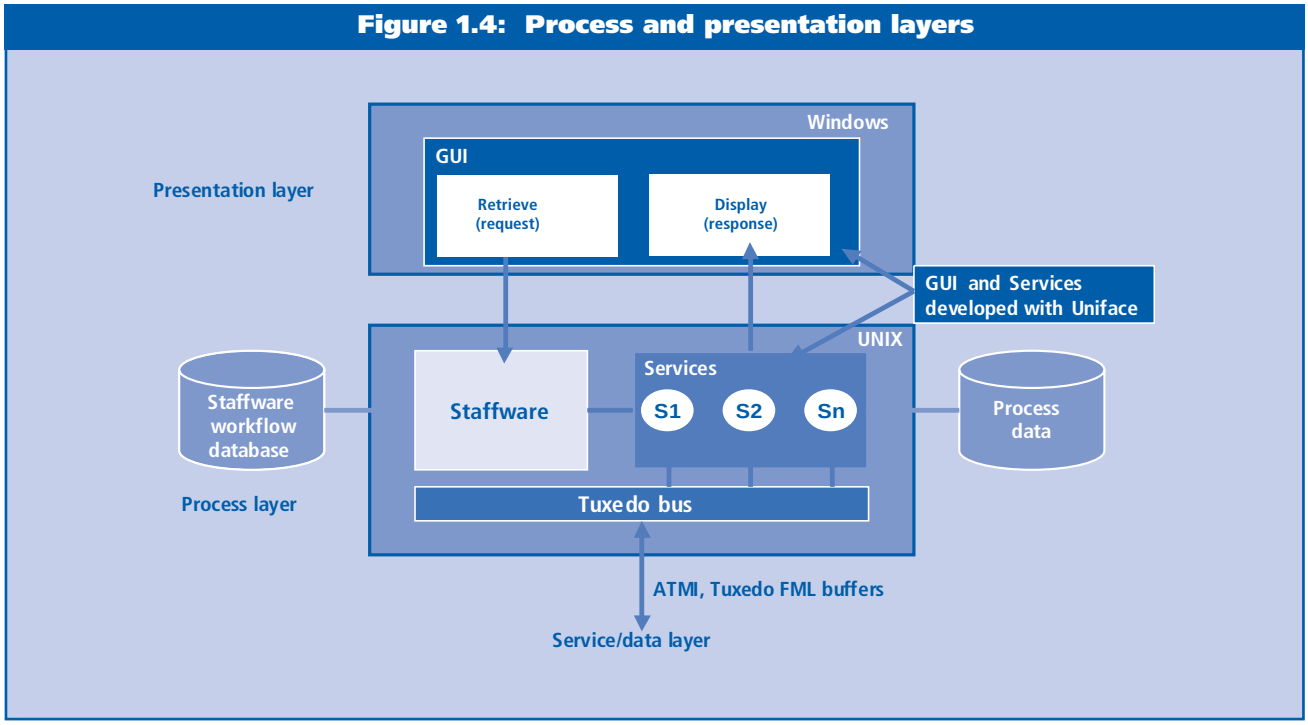
Implementation details: the Web interface for Vecozo

Web-based users coming through the Vecozo portal are also able to access Poseidon, for example to check if a person is insured with Agis. Increasingly, electronic claims are submitted through Vecozo, using SOAP. Figure 1.5 shows the structure.

The Vecozo system uses a secure connection to Agis, via HTTPS running over the Internet. After the Agis firewall, a Windows server running IIS receives the message and passes it to an application running under WebLogic Server (WLS), using HTTP.

This application then invokes Tuxedo services — using the interface between WebLogic and Tuxedo provided by BEA as part of the WebLogic family. This in turn calls the service layer to execute the logic and database access. The service components implemented in the service layer are invoked

Figure 1.4: Process and presentation layers



in exactly the same way as they would be from the workstation interface. They are unaware that the request has come from a different class of user.

Results and advantages of Poseidon

One of the business goals in embarking on the Poseidon project was to automate more of the work done by Agis, leaving staff free to focus more on their partners, such as hospitals. This goal has been achieved. In addition, the GUI interface has reduced training time to a fraction of what it was when users worked with a terminal interface.

There has been a substantial decrease in errors and reduction in the time taken to process claims. The result has been improved service levels and cost reduction:

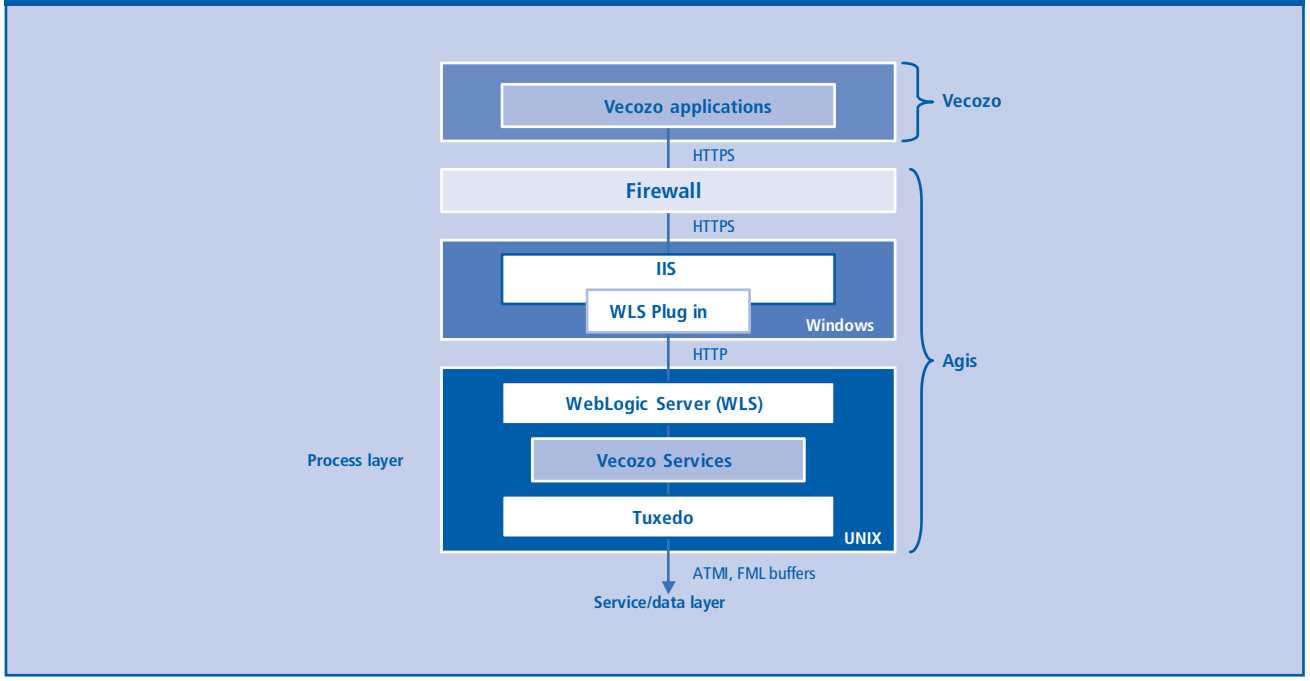
- **the number of outstanding work-in-progress jobs has decreased to under 10% of what it was**
- **the number of calls to the call center has been reduced to not much more than half of the former level**
- **the average time to process an application has been reduced to 2 days, down from 16 days**
- **75% of the work is now completed within a single day.**

From a system perspective, the separation of the definition of process into the process layer, with the implementation of the logic of the services as service components in the service layer, has resulted in increased flexibility:

- **applications are now easily adaptable**
- **new components can be added, increasing functionality**
- **maintenance time has also been reduced, because of the use of well-documented, self-contained components.**

The performance of the Poseidon environment is good, with fast response times. This was an important technical goal of the project, and has been achieved by careful design and product selection. The Tuxedo system in particular has proved efficient, as well as stable, as an integration vehicle. The benefits of easy integration with J2EE, in the form of WebLogic, have proved valuable. In other words, the expectations of this kind of service-orientated architecture, divided into well-defined layers, have been realized. The strengths of each platform have been exploited, resulting in a scalable application. By using an adaptation of the original IKAZ application, as the service and data layers, risk has been reduced, since the same logic has been extensively re-used.

Figure 1.5: External interface to Vecozo



Management conclusion

This case study from Agis describes the business goals, architecture and implementation of a complex integration project. An existing, somewhat monolithic, application running in a mainframe was adapted to a service-orientated architecture. A new superstructure was implemented. This now provides:

- ***users with a higher level of automation and ease of use***
- ***implementers with an environment where it is easy to adapt to new business needs.***

Risk was minimized by building on the existing and functionally rich and stable IKAZ application. Both the technical and business results justify this project in particular, and this kind of approach in general. Furthermore, these results were not achieved at the expense of performance, providing a useful indication for similar implementations.

The implementation of the project also re-inforced a number of good practices which, although they seem all too obvious when stated, are all too often not followed. They include:

- ***the complexity of this kind of implementation requires an initial use of short, proof of concept, projects to try out ideas and approaches with a minimum of investment; such projects are worth a great deal more than discussions around a whiteboard***
- ***the application is now spread over several platforms; it is especially important that the different parts of the organization work together to avoid difficulties when dealing with problems or changing the infrastructure***
- ***for the same reasons, change management takes on additional importance; care and effective procedures are necessary***
- ***the need to use skilled people on such a project needs to be emphasized; this sounds obvious but there is (sometimes) a tendency to think that sophisticated tools can replace skilled people — they cannot, even if they do make skilled people even more productive***
- ***finally, the project confirmed the importance of senior management support and realistic expectations when embarking on a project as ambitious and business critical as this.***

Are 'Enterprise Integration Buses' meaningful?

Charles Brett
President C3B Consulting and
President, International Advisory Board
MIDDLEWARESPECTRA

Management introduction

Enterprise Integration Buses are becoming ever more important, both within an organization (intra-enterprise) and beyond an enterprise (inter-enterprise). That importance is constantly being re-emphasized by the need to tie the many and diverse elements of business processes and information technology together in ways that benefit the whole business.

In that context, the objective of this analysis is to consider whether the concept of an Enterprise Integration Bus (EIB) is meaningful (and what it should contain to warrant the description). The approach taken by Charles Brett is to look at:

- ***the challenge which EIBs seek to address***
- ***what each of 'enterprise', 'integration' and 'bus' mean***
- ***layering in an Enterprise Integration Bus: business process, brokering and delivery***
- ***Service Oriented Architectures or EIBs or ESBs or application servers***
- ***should you approach from the bottom up or top down?***

The challenge

Few, if any, organizations in the 21st Century own only one application. Most have introduced (either by building their own or buying from third parties) different software applications that are dedicated to automating functions that are as diverse as:

- **human resources**
- **payroll**
- **order entry**
- **inventory**
- **general accounting**
- **MRP/ ERP**
- **R&D**
- **process monitoring and control systems (flow meters, gauges, specialist plant, etc.).**

Depending on the size and age of the organization, some of these applications may have been written in-house (so called custom-built applications) while others may have been bought in from third party application vendors like SAP, PeopleSoft/JD Edwards, Oracle and a host of other vendors (large and small). Since the 1990s, most applications run on their own dedicated hardware systems — rather than on the centralized, and shared use, hosts that were more common in the 1970s and 1980s.

In addition, sitting on yet more systems (particularly PCs) are still more applications, from personal databases and email through to specialty ones (like those used for process control or for handling specific activities like those found in production or research). The majority of these specialty applications are organization-specific; frequently it is these applications which support the activities that are unique to an organization.

Plus, increasingly there is the concept of the extended enterprise — where computing is being extended out beyond traditional office and manufacturing type buildings. Access to systems when mobile, whether in-house or external, is assuming an ever greater importance in the operation of successful businesses (of all forms, from manufacturing to finance to health care to ...). Similarly, there is an ever greater demand for 'systems' — that used to be outside of the scope of IT — to be integrated: typically these involve process, telemetry, building, network and other forms of control that previously have existed in 'parallel processing' environments.

One common theme running through the past 30 years is that few of these application systems (the underlying hardware, operating systems and the applications themselves) work with other application systems (whether inside an enterprise or with outside partners, suppliers or customers). Applications and their systems have, in effect, become islands of automation — automating specific functions but without the capability to interwork with upstream sources or downstream destinations. The result is a curious and contradictory collection of individually tuned systems which live in isolation from each other — much as if you had hands and feet and thighs and elbows which are not attached to the body.

From a business perspective this isolation is inefficient and expensive. While selected functions have been automated, that automation does not extend beyond the specific problem for which each application was written. The increase in demand for the inclusion of mobile and process control computing only emphasizes this further. Up until recently, the 'solution' was to use people as the means of integration (the 'middleware' to tie diverse applications together). People are still found — even in the 21st Century — printing out the results from one application in order to re-enter that information into a subsequent application.

If this seems absurd in a globally connected, mobile, Internet world, it is. Nevertheless, there remain many instances of high profile organizations possessing sophisticated Web sites (for example) to accept orders. These orders are printed out from the Web system — to be re-entered manually to the order taking system. The same applies to (say) meter readings; these may be obtained from a 'process control system', printed out and then re-entered into the 'commercial system'.

Ironically, the disintegration of processing is more prevalent than integrated processing. Indeed, because there are millions more applications in use today, there is less — relatively — integration than there was in 1980 or even 1990.

Business executives, when they understand this, do not like it. They know that an organization's ability to do business, to be competitive, is shaped by how well and how fast its various different functions (those that add the value) can work together. Or, in threat terms, those organizations which appear to be slower or less responsive than a competitor, will lose sales — even if their products or services are superior.

Furthermore, there is executive dissatisfaction with the lack of standards as well as the delivery of computing as a service to the business (rather than as a self-standing function). Standards matter as they offer a measure of future proofing. Services matter because they focus on addressing the business problem, rather than on technology itself.

Thus the business requirements that are driving a push for an Enterprise Integration Bus can be summarized as:

- **the need to make each enterprise operate better internally (this includes extending the enterprise to deliver mobile and process control capabilities); this encompasses the elimination of unnecessary isolation of function**
- **the desire to work with other enterprises**
- **the demand for greater adoption of standards and services, not least to insulate against unwanted change.**

Integration

Integration is about connecting different applications together, preferably with a minimum of rewriting of what already exists and is known to work and a maximal use of standards.

At this level, integration sounds simple. In practice it is not. Furthermore, integration comes in many forms. Successful integration occurs when a coherent, consistent, reliable and scalable approach which is manageable and secure is introduced. This must also be appropriate to the specific enterprise's needs: it is as easy to over-do what is necessary as it is to under-deliver.

In the context of an Enterprise Integration Bus, integration is about enabling existing (and new applications) to work together in ways that contribute to the business. While this concept is not new, delivery in the past has been rare. It no longer has to be — when sufficient thought has been applied. Enterprise Integration Buses have a role to play if they can address this need.

Enterprise

The enterprise, or organization, has become the focal point for integration. In the past, individual units within an enterprise (perhaps a subsidiary or a division or even a department) had the freedom to 'take their own IT initiatives'. This is no longer attractive, because the lack of co-ordination has created the very diversity that exists today. Executives expect the enterprise, and even the extended enterprise, to behave as one — so that the inefficiencies of isolation are overcome or avoided.

An enterprise's needs are commonly found in three forms:

- **first, there is the need for it internally to make its many (and diverse) applications interwork: this is intra-enterprise integration**
- **second, there is extending the enterprise beyond its traditional brick and mortar boundaries — specifically the exploitation of secure, mobile computing and or process control**
- **third, there is inter-enterprise integration; this is where third parties like customers, suppliers and partners become involved.**

An intra-enterprise focus is the most common target for integration. Ensuring that an enterprise runs efficiently is a target beloved of executive management. If an enterprise can achieve this it can deliver economies of scale and processing as well as react to changing circumstances with speed.

The inter-enterprise focus shares the same objective of integration. But this integration occurs across organizational boundaries, which compounds the challenge in several different ways. These include:

- **specific inter-organization issues (from security to making different applications work together to exploiting differing databases, networks and physical systems, for example)**
- **the existence of quite different orders of magnitude: for supply chain purposes, one organization might only be dealing with a handful of key suppliers but then selling to tens of thousands of small businesses or even to millions of direct customers (possibly connected over the Internet) while another organization might be the exact reverse — and there is every possibility in between.**

Mobile computing — in a sense a new 'wild card' in integration — applies to both. It can be as diverse as providing internal information outside traditional boundaries to delivery of new facilities to third parties (in an inter-enterprise context). This can make adoption that much more complex. Similarly, although with different implications, the inclusion of the process control dimension is rising in importance.

The key point to be made here, nevertheless, is that inter-enterprise integration can involve scales of integration which are far larger than intra-enterprise integration. Indeed, a distinction should be made between implementing inter-enterprise and intra-enterprise integration:

- **if you opt for an integration solution to satisfy intra-business needs, that solution will likely not be appropriate for extended or inter-enterprise requirements (for many reasons)**

- **conversely, if you introduce a solution for inter-business needs, that will almost certainly be sufficient for mobile or intra-enterprise requirements as well.**

The message here is: think broadly before taking a decision about focusing on an intra-enterprise only solution.

One further point should be made about the description 'Enterprise'. An Enterprise Integration Bus must be capable of handling the whole Enterprise and beyond (if third parties are to be involved). But that does not mean that the first steps have to embrace the whole Enterprise from Day One.

Having the vision and knowing what will be needed by the whole Enterprise matters when drawing up the plans for an Enterprise Integration Bus. But this does not mean that everybody and everything has to start at the same time. Gradual steps are preferable, once the essential logic has been put in place.

Bus

In the information technology arena, terms like 'bus' and 'hub' are used to describe particular configurations or topologies. But they can also be used to 'accuse' one side or the other of having (or not having) certain characteristics.

In reality, there are minimal practical differences between a Bus and a Hub approach. If you have two or more Hubs you effectively have a Bus. If you have one Hub you have a Bus (Figures 2.1-2).

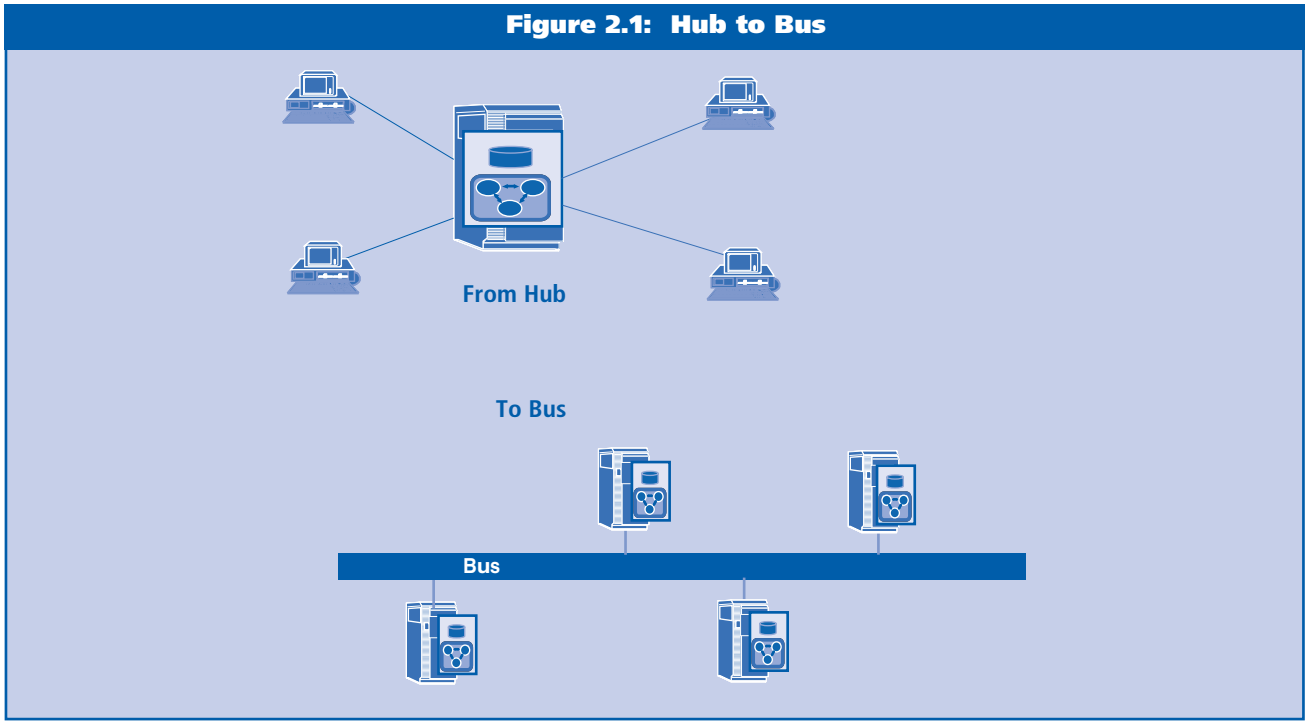
Much energy has been expended in trying to create artificial sales advantage by 'describing' one side or the other of having a 'Bus-only' or Hub-only' approach. This 'differentiation' is nonsensical (although see Tom Welsh on page 32 arguing the opposite).

What matters in the context of an Enterprise Integration Bus is that an appropriate topology can deliver the integration needed across and, if relevant, between enterprises. Thus, one could select an Integration Hub approach or an Integration Bus approach: both apply and either will only be as successful as each one's relevance to its sponsoring Enterprise(s).

What is an Enterprise Integration Bus?

The most obvious way to approach the description of an Enterprise Integration Bus is to describe a host of technologies that should be included. This is not necessarily a helpful way to approach a complex subject — not least because any selected technologies themselves tend to shape the end result. A preferable alternative is to consider function, and what an Enterprise Integration Bus needs to provide if

Figure 2.1: Hub to Bus



it is to be meaningful. For example, it is practical to describe an Enterprise Integration Bus as being characterized by its relationship between three 'layers' of:

- **business process**
- **broker function**
- **delivery.**

Business process

The highest level is the first — the business process. Business processes are the combination of activities within an organization that deliver results and add value. Examples include taking an order, manufacturing a product, delivering a service, executing a foreign exchange deal, producing the annual (or monthly) accounts, etc.

What characterizes all of these are the following:

- **people are involved (not just system to system or application to application activities)**
- **multiple sub-steps (or sub-processes) exist within each business process**
- **these sub-steps frequently transcend multiple sub-systems**
- **most organizations need to be able to adapt or create new business processes rapidly, if they are to remain competitive (and delete them, as required).**

Manifestations of business processes come in products with names like 'process flow', 'work flow' and 'choreography' or 'orchestration'. These tools provide the means to describe and then co-ordinate the complex activities — that might be short or long lasting, or a combination of each — which make up a business process.

With all this said, another distinction should be made. Somewhat confusingly, many will (correctly) say that applications, or application suites, embody business processes; after all applications do encapsulate business activities. But the business processes where an Enterprise Integration Bus might apply span individual applications (or application suites). Business processes in the context of an Enterprise Integration Bus are about:

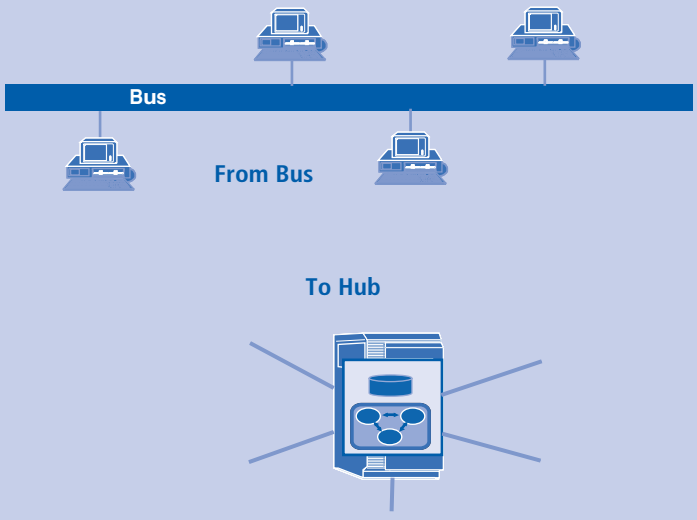
- **removing the isolation of individual applications (and application suites)**
- **making these part of a larger, extended whole.**

Delivery

At the opposite extreme, and arguably at the lowest technical level, is delivery. In all organizations delivery is critical. In information systems, there is a need to carry information from one activity (be this application or database or process or whatever) to another.

This is often referred to as the transport of information or

Figure 2.2: Bus to Hub



events. It is akin to the postman collecting letters (messages) from multiple sources and delivering these to designated destinations. In different circumstances different qualities of service are needed by different processes — akin to next day delivery, registered post, poste restante, guaranteed delivery, insured delivery, etc.

To complicate matters, although this also provides diversity and choice, there are many flavors of delivery (transport), including:

- **free form text (email, short message service or instant messaging) and fixed format (where the data is in a predetermined or known form)**
- **synchronous and asynchronous — the key difference being whether two or more resources all have to be available for delivery to occur: with asynchronous delivery, the simultaneous availability of another resource is not a requirement (just as the recipient of a letter does not have to be present when the sender posts it)**
- **persistent and non persistent: the former determines whether information being carried should be 'copied' until delivery has been confirmed while the latter does not offer this possibility**
- **point to point and publish and subscribe, where the former is about A to B or C or D while the latter possesses a publisher who publishes to subscribers (who may not be known to the publisher) who choose to subscribe**
- **real time and decoupled, where the former is about speed of delivery while the latter may be less concerned with immediacy and more about confirmation of delivery**
- **transactional and non-transactional, where the former enable business transactions to occur in a clearly understood environment (either the delivery has occurred or it has not) so that the 'state' is always clear (or this may not matter, as in non-transactional)**
- **reliable and non-reliable; the choice of reliability and non-reliability will depend on circumstances and needs — and, at different times, either may be applicable**
- **heavyweight and lightweight; this is not a measure of worth but more a measure of what is being carried and its relative value; the possibilities are immense, from the simple (lightweight) — such as a 10 character reading from a meter on a gas pipeline or on an oil tank — through to the delivery of a complete bill of materials — that is 100MB in size — for a machine with all its associated sub-components (heavyweight)**
- **programmatic and non-programmatic, which cover different ways or styles of achieving delivery.**

Furthermore, the descriptions above are not mutually exclusive. They intertwine and mix. It is as possible to have, for example, decoupled, heavyweight transactional point to point messaging as it is to have non-reliable, non-transactional, lightweight, publish and subscribe delivery.

The point is that delivery requirements come in many forms, many of which have existed in isolation from each other. Bridging such isolation of delivery methods, by enabling differing forms of delivery to be combined and exploited, is a necessity in most enterprises — but not one that many have been able to expect, at least until the arrival of the Enterprise Integration Bus concept.

Broker function

In between process and delivery lies broker function. This has many aspects which can include:

- **switching/routing (which in turn can be address or content dependent as well as offer connections between different delivery forms)**
- **transformation (for example, between different formats or even languages)**
- **database updating**
- **enrichment of information (using data from one source to obtain additional data from other sources to create something richer than the original information — for example, using an Account Number to look up and then add the Account Name and address as well as FedEx details).**

The underlying concept is simple, and akin to a travel agent or insurance broker or foreign exchange broker. Brokers are intermediaries which co-ordinate the delivery of the correct combination of flights, car rentals and hotels or acceptance and issuing of insurance policies or switching between currencies. A broker between applications is intermediary software which mediates between the various different applications so that these applications can inter-work. Typically, they achieve this in several stages: by transporting messages, interpreting messages (although not in a pure linguistic sense) and routing them to appropriate destinations.

In addition, broker function can be used to reduce 'application spaghetti', via one or even multiple hubs. One or more brokers use delivery transports to convey information between applications, and can even provide degrees of externalized processing.

The point about broker functions is that they intermediate between other activities. Brokers can take an input of some form and then enhance or redirect it (depending on what actions have been defined); the altered output can then be delivered to one or more new destinations.

The significance of the broker function is that it enables automated actions to be located outside the source and destination applications. This decoupling of the sources and destinations facilitates integration, because much less has to be changed at those sources and destinations. In turn, this accelerates delivery and introduces flexibility — without losing or endangering what works already.

Complementary, if orthogonal, requirements

If these are the basic functions, there are additional complementary — if orthogonal — ones that must also be considered. These include:

- **security (which is critical in an inter-enterprise environment because few organizations are going to permit themselves to be held hostage to others' security implementations)**
- **administration/management; if an Enterprise Integration Bus cannot be managed it is as useful as an army without a commander**
- **availability/robustness; Enterprise Integration Buses swiftly become 'business critical'; 'losing' them is not acceptable**
- **scalability; the ability to increase or decrease capacity to suit changing circumstances is essential if business activities are to be sustained**
- **building (or 'development') tools; the capabilities of these will determine how easy it is to deliver function (at any level) as well as how easy it is to make changes.**

In broad terms, therefore, an Enterprise Integration Bus is a complex collection of different functions and capabilities covering delivery, brokering and more. All these must not only operate together but must be secure and manageable. Without the latter no Enterprise Integration Bus will be worth much, especially if the expectation is that it will become mission critical.

Services Oriented Architecture or EIB or ESB or applications servers

Indeed, in the past, some large organizations have created their own Enterprise Integration Buses — as an implementation of an in-house Services Oriented Architecture (or SOA). This raises some interesting issues in today's market place.

There are those (see Steve Craggs on page 20) who argue that:

- **an Enterprise Integration Bus (or Enterprise Services Bus — ESB) is a pre-packaged form of SOA**
- **what matters is that the function of an EIB or ESB lies in the combinations of products and services put together so that the customer effort to define an SOA is not so onerous.**

In this view, an Enterprise Integration Bus, or Enterprise Services Bus, is substantially more efficient than defining and then creating your own SOA. There is more than a germ of truth to this. Those organizations which have pursued the SOA route have taken 5+ years to deliver an SOA that has become a working reality. Few enterprises, or executives, have the patience (or resources) for this.

What also matters here are standards. Application servers play here. With their basis being founded on J2EE or .NET, these are capable of delivering a services-based integration solution (albeit in a form that requires much development and programming).

Nevertheless, standards do make interworking easier. They also provide some degree of insulation from unintended change (if something is built to a standard, then replacing one part with another build to the same standard should be less problematic than having to create a new combination). Standardization in an Enterprise Integration Bus can (some will argue 'should') manifest itself as a service — which can then be exploited for the enterprise's benefit. Going further, standards also make the delivery of inter-enterprise function simpler.

This dimension to an EIB is producing new initiatives, most notably from the Web Services and even the Grid communities. Indeed, it may be that future EIBs will just be accumulations of Web or Grid Services which can interoperate across platforms. Certainly the concept is a valid one. It is, however, some way off — and 'integration waits for no enterprise' (to paraphrase).

Bottom up or top down?

A dimension which is often ignored is how to approach integration in the form of an EIB or ESB or whatever. Put simply, the issue is one of whether to start from the top (with business processes) and work down or to work up from the bottom (from delivery).

Most integration vendors prefer to start at the top and work down. Why? There is direct involvement with decision makers (which almost always means larger, longer sales). The scope is greater. More possibilities are promised. The discussion is about business processes and what adds value to the business.

In many senses this is utterly logical. Taking the big view and then decomposing this into its smaller, constituent elements sounds right.

Unfortunately this does not map well onto the way organizations buy products or even into concepts. In addition, integration is about linking what already exists. Attempting the top down runs the significant business risk that 'what is desired' will be imposed on 'what can work'.

The attraction of a bottom up approach is not only that one can use the foundations of what exists, and extend these. It is also inherently safer (with less business or technology risk) and easier to initiate.

In part, this is a consequence of how organizations work. The top down approach is excellent if there are all of: continued executive support plus excellent project management plus supporting integration solutions plus the willingness to invest (both financial and in staff resources). Perhaps regrettably, such combinations are rare.

In the absence of these preconditions, starting from the bottom up is more practical. Much more may be achieved with considerably less.

The relevance to an Enterprise Integration Bus (but less to an Enterprise Services Bus or an application server approach to integration) is that it suits bottom up to top. One can start with delivery and flow up to brokering and even then to process control. Equally, and often not thought of, it is almost certain that most top down approaches will need both delivery and broker functions.

In this light, an Enterprise Integration Bus may not necessarily be the most exciting or interesting 'thing' for a vendor to sell. But it may be a more practical approach for an enterprise to deploy. It is a logical building block that does not preclude more extensive process or work flow orchestration and choreography in the future.

Management conclusion

Are Enterprise Integration Buses a meaningful concept? This analysis argues that they are, for several reasons:

- ***they can offer integration across an enterprise, and even between enterprises***
- ***they provide delivery, brokering and the initial stages of process co-ordination are part of an ESB***
- ***they are relevant for a top down approach but, more significantly, they facilitate a bottom up approach — building on what already exists***
- ***they bridge the extended enterprise, including the mobile dimension, and the process control dimension (areas that previously have existed outside traditional IT).***

Does this mean that EIBs are simple to implement? Not necessarily. But, by comparison to the effort required to create, and then make manifest, a Services Oriented Architecture or application server or Web/Grid-based integration, the relative simplicity makes for greater utility. In the end, of course, it is the results which matter.

Raising EAI standards

Steve Craggs
Principal, Saint Consulting and
Vice-Chairman, EAI Industry Consortium

Management introduction

Over the last 10 years, business integration technology has become vitally important for many companies — because it applies to a number of key business initiatives, such as:

- ***mergers and acquisitions***
- ***eBusiness and the Web***
- ***Customer Relationship Management (CRM)***
- ***Straight-Through Processing (STP)***
- ***Supply Chain Management (SCM)***
- ***legislation, such as industry deregulation and 'T+1' trading.***

The market for this integration technology — most commonly referred to as the EAI (Enterprise Application Integration) market — has changed significantly over the years as have business conditions and needs. In this analysis, Steve Craggs discusses how it promises to change even more in the future as EAI continues to become more and more established as mainstream IT technology. As he shows, initially the EAI vendors pretty much controlled the market, dictating the market requirements. Gradually the increasing surge of EAI implementations has resulted in more and more attention being paid to the business needs of the end user organizations investing in this technology.

Setting the scene

Ever since its inception in the mid-1990s, the EAI market has seen dramatic growth as more and more organizations start to understand and appreciate the value of better integration throughout their operations. The concept behind it is essentially very simple: by improving communication between different applications, systems and environments it becomes possible to achieve higher levels of automation, efficiency and quality of service throughout the fundamental processes that drive each business. Furthermore, new applications can be brought to market faster and hence deliver results more quickly.

As the millennium turned, the widespread economic downturn forced more and more companies to focus on the bottom line. This has provided a second major driver for improved integration. The attraction is that of improving the return on existing investments rather than supporting the cost and effort of delivering new ones. Streamlining and automating existing business processes, both internally and across the entire corporate value chain, offers a reduction in operating expenses and improves profit margins. It is no coincidence that — even through the difficult economic climate of the last few years — enhanced integration has been at the top of IT priority lists at many organizations.

The EAI market has also changed considerably throughout its evolution. The market has suffered from hype and manipulation, with new concepts and ideas being propounded at increasing speed by vendors and analysts keen to raise their profile in what seems to be an attractive market place, sometimes with scant regard for users' actual needs. The resulting confusion has caused many companies to stumble, often bewildered by a cloud of marketing and over-zealous sales activity.

At the same time, as the market matured, standards have started to emerge. This has been beneficial because these have brought a level of clarity to the existing confusion while at the same time drastically changing the EAI cost/benefit model. These standards:

- **are themselves now maturing**
- **becoming more widely adopted**
- **fueling a new collection of standards-based EAI vendors that operate with less of the investment overhead of the incumbents.**

To understand the state of the market and the direction it will take over the next few years, much can be learned by observing the dynamics of the market from its inception. This includes looking at:

- **the market taxonomy covering the different segments of the technology**
- **the extent of adoption of that technology**
- **the changes in the competitive approach to the market as a whole.**

Market taxonomy

One aspect which has changed significantly 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 layer to enable packaged applications — such as SAP, Siebel and PeopleSoft — to link more easily with legacy and newly developed Web and workstation-based applications. This new technology integrated these enterprise applications together — that is, it offered Enterprise Application Integration (EAI).

At this stage, EAI consisted of a MOM communications pipe, adapters and connectors with which to access source and target components, transformation engines to map information formats between sources and targets and some intelligent routing capabilities. If MOM was Phase 0 in the EAI marketplace, the packaging of this list of functions into 'message brokers' was Phase 1.

The early EAI market was characterized by infrastructure sales consisting of highly-priced software (due to the small number of competitors) and substantial associated consulting costs — the latter often amounting to five times the cost of the underlying integration software. Two developments then occurred that caused a number of major transitions in this early marketplace:

- **first, competition intensified, with new players constantly entering the market**
- **second, more IT budgets were moved under business unit control.**

EAI vendors were quick to realize that, to sustain revenue growth, they would have to climb the value stack as quickly as possible to differentiate their offerings from their competitors' offerings. This meant moving to a more business-oriented value proposition. All of a sudden the talk in the marketplace was about:

- **business process integration (BPI)**
- **business process management (BPM)**
- **business process automation (BPA)**

with the story being one of business process flowcharts created by business analysts that were somehow mapped down to the underlying IT components and programs. Some vendors actually did seem to understand what business processes were all about. Others, however, were guilty of 'putting lipstick on the pig'— making cosmetic changes in order to be able to claim some sort of business process functionality.

A third cause of transition in the EAI market was driven not by vendors but by users, the people actually implementing EAI solutions and moving them into production. Many user companies found that there were major holes in the product offerings they had bought when these were released into 'real-world' usage. Typically migration, co-existence and operations support in a production environment were found to be wanting. The production tools required by most installations, such as monitoring tools, were considered by too many EAI vendors only as an afterthought.

The assumption on the vendors' part was that once everything was engineered to use the new EAI technology then life would be fine. In reality few companies could afford (or want to introduce) a big bang approach to a new solution. The business risk was too great. The old must live with the new for a number of years. This reinforced the need to continue to support older, proven means of lesser integration including, for example:

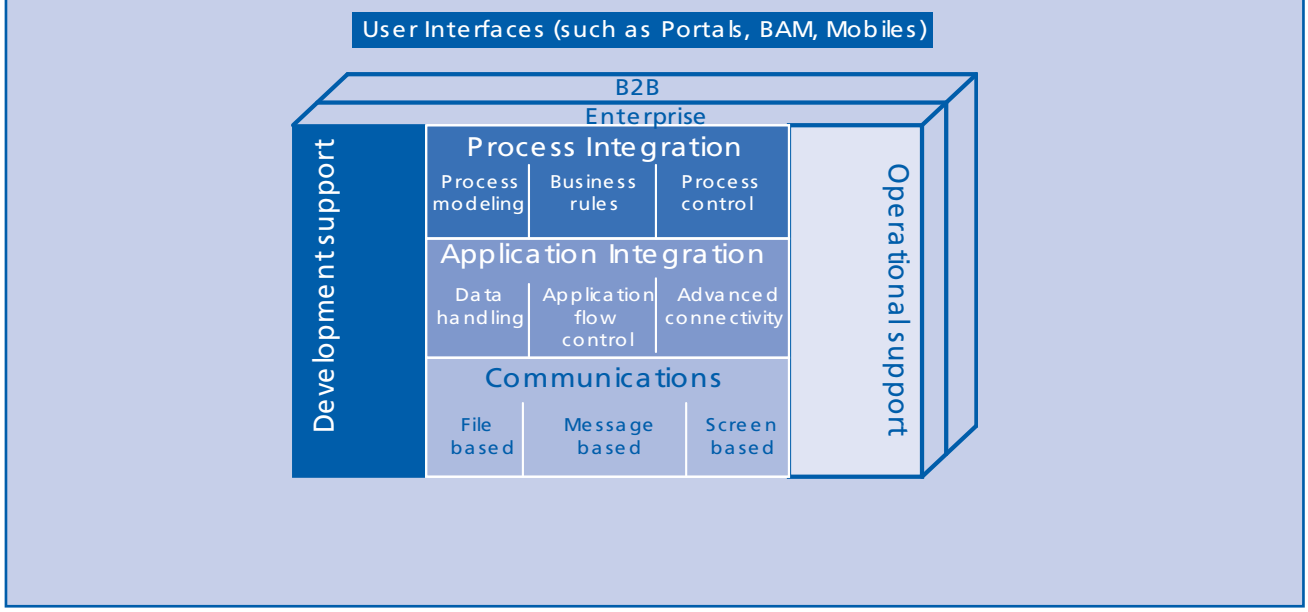
- **file transfer**
- **screen-scraping**
- **data integration.**

Today's marketplace

The cumulative result of these different influences on the EAI market can been seen in the modern view of the marketplace (Figure 3.1). Companies that want to implement an EAI strategy now have to consider each aspect of this taxonomy to determine the best approach. Generally this involves working with more than one vendor — not least because many EAI vendors are reluctant to spend resources trying to address the bottom (lower value) layer: they prefer to leave this to third party implementers or use partner offerings. It is the higher layers (with apparent higher value) that attract most of the EAI vendors' attention — due to the apparent linkage of these layers to business value.

Figure 3.1 represents Phase 2 of the EAI market. But does this mean that companies looking to implement integration solutions should be analyzing the solutions offered across all these segments? Should the support a particular vendor gives for business processes be as important a consideration as support for file transfer? The answers are that, although all of these segments might be of interest to an implementer in the long term, in the short to mid term there are probably other factors to consider.

Figure 3.1: A modern view of the EAI market place



Market Adoption

The EAI market has also been subjected to an enormous amount of marketing hype. It is important, therefore, to cut through this and obtain a clear picture of where the market actually stands today — not least because vendors continue to push their value propositions 'up the stack' to:

- **increase their leverage with the 'business' decision makers**
- **sustain product (and service) price levels.**

Lay people can be forgiven for thinking that the most important consideration in purchasing an integration solution is that it supports business process management effectively. However the reality is that it is the lower layers that are of more importance today. Furthermore, this is likely to remain true for the majority of EAI customers for some time into the future. Figure 3.2 offers a simplified representation of the market adoption position today.

In the current economic climate, return on investment (RoI) is essential. Payback times need to be rapid. This has created a bias toward the simpler forms of EAI over high-function infrastructure solutions. Integrating with legacy systems is also a struggle, with only the major applications receiving focus from solution vendors. Specialized solutions such as industry-vertical specific or custom developed ones tend to still need substantial work to integrate successfully. This in turn exacerbates the need for expensive technology specialists

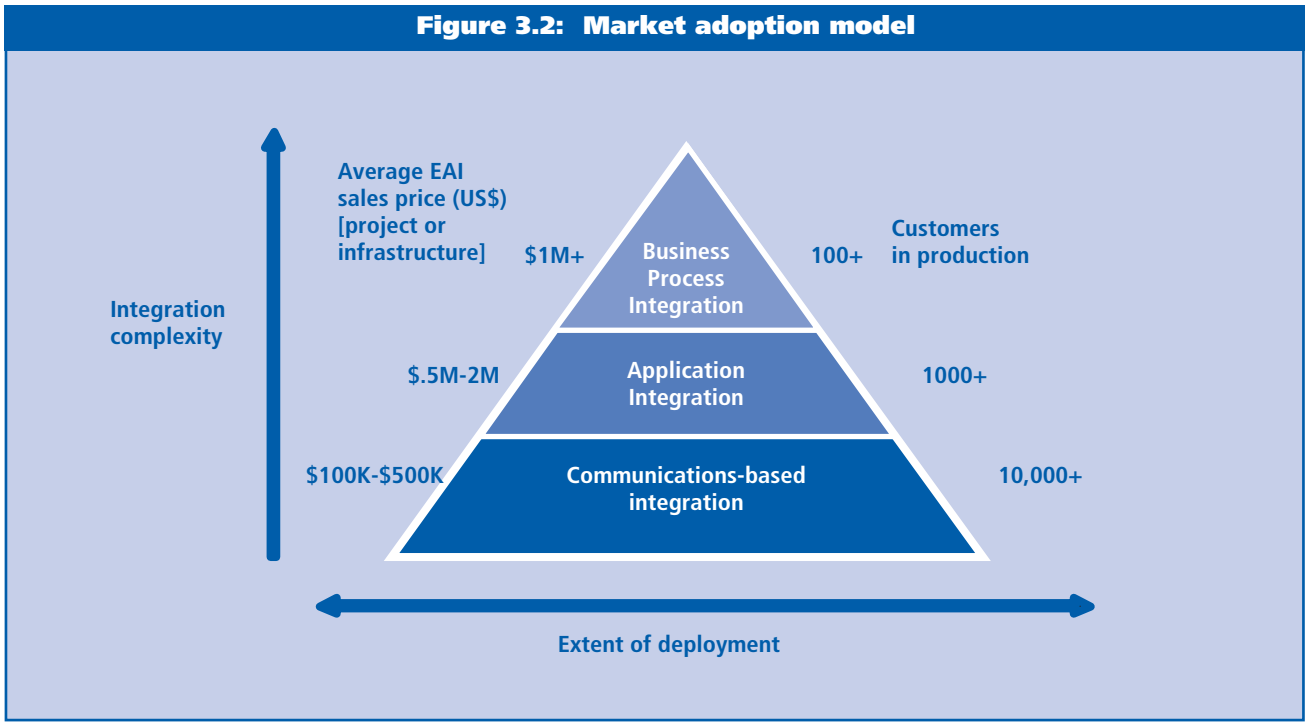
Finally, the whole aspect of business risk can present a problem for EAI planning and implementation. Obviously, any implementation of new technology comes with an associated level of business risk. But risks are magnified in the case of EAI by several factors, including:

- **EAI technology has traditionally been complex, and hence prone to error**
- **many EAI vendors are relatively young and/or small, which introduces the risk of lock-in combined with uncertain vendor longevity**
- **future application integration needs are still likely to require extensive custom coding, resulting in quality, support and time-to-market issues**
- **integration projects typically rely on external consulting services (often far outstripping the software in terms of costs) rather than utilizing existing in-house resources and building a skill base in the new solution.**

New (and old) challengers

From the above discussions the general picture is of an EAI market where EAI vendors were luring customers up the value stack towards the promised land of

Figure 3.2: Market adoption model



full-scale integration of every business process that is fundamental to an organization's operation. Although the pace along this journey is not nearly fast enough for the EAI vendors, most everyone agrees on the broad approach and the steps on the journey ... Or do they?

The answer is, of course, No. As often happens in business, when there is a market with large revenues and strong year-on-year growth, there are new challengers who are trying to achieve market entry.

The last few years have seen still more transitions in the EAI market as a number of alternatives, to the accepted approach, have gained credibility. In increasing order of importance, three of the primary alternatives are:

- **work flow vendors entering the EAI space from above**
- **old technologies being revived on a pragmatic basis**
- **solutions being built around emerging standards.**

When the EAI marketplace expanded to cover the concept of business process management — beyond the more technical requirement to interface to different packages and solutions — it offered a bridge that opened up an opportunity for work flow vendors to enter the picture. These companies had built their reputations on being able to work at the business level with clients, helping them to organize their business processes and work practices in a more automated and efficient manner. By adding some basic EAI technology to their solutions, either developed internally or through obtained via partnerships, these work flow vendors are now starting to put pressure on the EAI market by threatening to unseat traditional EAI vendors.

The second threat came with the resurgence of traditional technologies. Although the hype surrounding EAI managed to persuade many buyers that this type of technology was all new, it has been around for some time. Given the weaker IT investment cycles forced on buyers through the general economic world situation over the last few years, organizations have re-examined existing investments — in such 'antiquated' technologies as file transfer and data integration — and found them good.

However, the biggest challenge to the incumbent EAI leaders has been the emergence of standards within the EAI market place and the corresponding entry of nimble, fleet-footed vendors. These are offering standards-based solutions with support for key environments such as the Internet and application server platforms.

The standards effect

Software markets tend to follow a basic pattern. As a new market is created by the development of a new, viable technology (assuming the technology has real value, of course), a host of vendors both new and old start to develop and offer products based on the new technology. These products implement the technology in whatever manner their builders see fit — based on the requirements perceived by the vendor in question and the creativity and expertise of the individual development teams. The result tends to be a set of proprietary solutions, with pricing dynamics based on the extent of investment in the building of the new technology/product combination.

Next, assuming a market thrives and grows, the standards issue starts to appear. There are several reasons for this:

- **proprietary vendors with existing market share strength are always keen to see their own implementations declared as the standard, thereby giving them a competitive edge**
- **new entrants to a market want standards to make their development easier, better defined and lower cost**
- **standards bodies normally contain technical evangelists who see it as their personal duty to create what is 'best' for an industry or segment**
- **end user organizations prefer to minimize (or avoid) vendor lock-in if possible, since this reduces both risk and prices.**

Somebody somewhere defines standards. This is the critical point in the software market development.

One of two things then happen at this stage:

- **either the standards gain a high level of support, thereby forcing proprietary vendors to adopt them to remain competitive with new entrants**
- **or the standards are ignored by all but the most ardent fanatics.**

Usually the latter outcome happens when one technology implementation has dominant market share and becomes the de facto standard, more by presence than any other form of industry support. Another possible inhibitor to standards acceptance is that standards cover only a subset of the available functionality in proprietary implementations. This is not unreasonable since, to gain acceptance, it is important that the standard satisfy the 'lowest common denominator' of functionality implemented in the majority

of proprietary solutions. The issue is then one of how well this subset of functionality is able to satisfy market needs.

An obvious example of this is TCP/IP, accepted for some time now as the ubiquitous standard for intercommunication. In fact the 'real' standard — as defined by one of the standards bodies — was OSI/TP, but hardly anyone implemented 'the standard' because TCP/IP already had a massive presence.

Standards have been appearing in the EAI market over the last few years. The key question now is — will these be ignored or are they here to stay?

Standards in the EAI market

The EAI market in its broadest sense covers all aspects of business integration. This is a market where standards logically should make an enormous amount of sense.

Some of the generic drivers for standards have been touched upon already, and of particular relevance in this market is the avoidance of vendor lock-in. Many of the EAI vendors are companies created solely to serve this market. Most are either making heavy losses or only just approaching profitability. In addition, the EAI market is seeing much consolidation with numerous acquisitions over the last five years.

These factors raise the question of vendor longevity. The danger of being locked in to a proprietary vendor technology represents a considerable business risk (one has only to look at how the State of Connecticut reacted to the Oracle offer to buy PeopleSoft to understand the degree of concern).

Furthermore, there are special considerations for standards in the EAI market, stemming from the requirement to link together systems across the entire value chain:

- **the need to integrate processes across third-party as well as internal systems**
- **the need to be able to connect to any or all of the internal applications and packages**
- **the need to support the various transaction processing environments such as TP monitors and application servers**
- **the need to potentially offer services for external usage**
- **the complexity of EAI technology, and therefore the increased level of opportunities for 'black-box' functionality.**

There has, therefore, been considerable activity in the standards arena around EAI. The first development to affect the EAI space was the emergence of XML as a self-defining way of formatting data. Instead of requiring every company or department to implement a proprietary set of data formats, a standard way can specify the formats that can be understood by any other system. XML has quickly become the way of dealing with information flowing between different parts of the business process and has considerably reduced the extent to which transformations are required.

With the popularity of Java-based application servers as a way to build modern applications, many EAI standards have stemmed from the Java movement itself. For example the JMS (Java Messaging Service) specification is implemented by thousands of companies. JMS has gained such a strong mind-share in the market that other MOMs such as IBM, in its WebSphere MQSeries Enterprise, felt it had to offer a JMS implementation. The secrets to the popularity of JMS implementations are indicative. They combine:

- **a level of functionality suitable for almost all market needs**
- **an attractive price point**
- **the security of (virtual) vendor-independence since JMS is a defined standard API.**

The J2EE Connector Architecture specifications has gained significant traction, largely because of its intention of providing a standard for interfacing to applications. When the specifications are completed and enough large application packages implement them, JCA will also remove a considerable amount of effort in the integration implementation cycle. The development of custom-built adapter code to integrate with in-house packages and applications is often the largest, most complex and error-prone part of any EAI implementation. Once a project has been delivered it still incurs considerable maintenance costs.

Finally, Web Services collectively covers a range of standards specifications (SOAP, UDDI, WSDL) that are aimed at easing the entire job of integrating applications and business processes together. The idea is to provide a standards-based way for business services to be described, published, discovered and accessed both within and across organization boundaries.

Supporters of Web Services predict a future of globally available business process or sub-process services, accessed through a global directory and usable by any organization anywhere. However today's reality is less than this

although it is likely that Web Services will be used internally within private networks for several years.

Standards make a great deal of sense in the EAI space, and as a result they have established a considerable presence in the market already. Virtually all new RFPs issued by prospective buyers of EAI software have a section on standards implementation and/or direction. So how has the emergence of these standards affected the market and what is likely to happen in the future?

The impact of EAI standards

The XML standard has generally been embraced with open arms by both EAI vendors and end users. Most EAI offerings now include a transformation engine to map the data formats used by the source application to those used by the target application. In practice, what most EAI vendors do is:

- **map the source data into an XML format**
- **pass the information around internally in this format**
- **move from XML to the target data format.**

This is a much more flexible approach and is easier for vendors to handle. From an end user point of view this is satisfactory because, as more data formats are switched to an XML format, they can just be plugged straight into the EAI tool. XML offers a win/win for both vendors and end users.

The Java-based standards (JMS, JCA, etc) have also achieved considerable mind-share in prospective buyers' minds. However, unlike the XML standards, they have encountered resistance amongst the proprietary EAI vendor community. This resistance is not always in the form of direct opposition. But it is clear that some of the early EAI vendors are not fully behind these standards even though they may appear to pay lip-service to them.

The first issue is that these early vendors have created their own proprietary solutions to the EAI technology problems, and software developers are notoriously biased when it comes to alternatives to their own designs. They try to hold a position that standards fall functionally short of the proprietary solution in question.

However this argument misses the point entirely. Standards focus on addressing generic market requirements. Provided they do this, effectively any additional functional 'bells and whistles' are irrelevant. The vendor-internal business reasons behind the concerns of the proprietary EAI players go much further to explain resistance to standards:

- **adopting standards will be costly to the vendor in time and money**
- **many of the vendors' competitive differentiators are perceived to lie in their own unique implementation; standards level the playing field**
- **standards may make it easier for a customer to switch to a competitive offering (that is, vendor lock-in is weakened)**
- **because existing customers might already be using functions not covered by the standards, vendors have to support both old and new designs and interfaces.**

A secondary truth is that standards appeal strongly to a different breed of EAI vendor. A number of more recent EAI vendors have gained significant market traction by taking advantage of standards specifications to produce almost completely standards-based EAI offerings. These vendors have already built up impressive lists of large-scale successful implementations of their standards-based offerings and are seizing a growing share of the EAI market.

In addition, a number of application server vendors who have had to rely on partnerships (for proprietary EAI solutions) have become aware that they can now offer basic EAI technology cheaply without having to spend years investing in complex design. However this latter group of vendors (for example, BEA Systems) is unlikely, in general, to develop leading EAI components since they have many other areas on which to focus. Instead it is far more likely that they will embed standards-based EAI solutions from the 'new breed' standards-based EAI vendors.

The standards-based EAI vendors gain the benefit of operating on a very different financial basis from the proprietary vendors:

- **first, the standards definitions bound the development challenge quite tightly, avoiding the need for swathes of function that may or may not be of interest to most customers**
- **second, since the standards definitions are implemented widely by definition, they tend to raise the overall quality of the solution being built when compared to a completely custom approach**
- **third, the starting design point for these vendors is standards-based, as opposed to the situation for the proprietary vendors where supporting the standards often requires a force-fit of a proprietary approach with standards 'simulation' bolted on.**

The result is that the new EAI vendors tend to have a more efficient and productive implementation since they do not have any old interfaces or design assumptions to maintain. There is a further important point to be made here on competitive differentiation. It may appear that all standards-based offerings will be much the same, since they all implement the same standards.

This is, in fact, not the case at all. Standards definitions tend to cover the interface specifications at a technical level. In other words they cover the 'what' that has to be done, but not the 'how'. In turn, this leads to one of the major areas of differentiation for standards-based vendors — the implementation itself. So for standards-based vendors this means that investment can be focused on such key areas as ease of use, performance, scalability, security and indeed the overall design used to achieve the required functionality.

Taking these points together, the summary result is that the new standards-based vendors offer EAI products that generally:

- **offer a considerably lower price point**
- **implement standards in an optimal fashion**
- **rely on the quality and effectiveness of implementation as differentiators**
- **demand a more generally available set of skills**
- **can be substituted, if required, with relatively little effort.**

Standard approaches

So far, consideration has been given only to technical standards — that is, standards largely covering interfaces and protocols. However there is another very important area of development in the EAI market, which is not so much focused on the mechanics of how one component interfaces with another but instead covers the way that the integration task is carried out (sometimes referred to as the integration architecture model).

As EAI has matured a number of specific approaches to integration have emerged that are fast becoming standard, although with a lowercase 's' (not defined by any official standards body). One of the key ones is the service-oriented architecture (SOA) approach.

In an SOA model, the principle is that separate business operations or process steps are packaged as much as possible as self-contained components or 'services'. This has strong similarities to the 'old' object-oriented programming concept but, instead of the encapsulated component being a program or a piece of code, it is a set of pieces of functionality making up the service in question. In addition, a services-oriented architecture model frequently operates on an event-driven basis, very much as a traditional business process does.

The SOA approach has some fundamental advantages when applied to integration solutions:

- **a high degree of re-use**
- **the raising of the overall quality of service for the new solution**
- **the provision of a highly flexible and adaptable implementation**
- **the enabling of IT operational flows to reflect more closely the way an organization operates.**

Every business operation in an SOA environment is broken down into the basic set of services, which in turn form building blocks that can be used to assemble new solutions. The connectivity and integration of the components within these encapsulated services, and indeed of the services themselves, is provided by the EAI technology. Once the business services have been created, these can be viewed as self-contained black boxes with well-defined inputs and outputs. It is that which delivers the flexibility, adaptability, quality and re-use.

In an SOA environment it becomes a simple matter to switch a particular service for a new one as part of an overall business process. Indeed, because of the power of EAI, the new service might not even be internal but could be provided by a third party as an alternative.

Past developments around the SOA concept over the last few years are proving fruitful. They are already influencing the direction of the EAI market. Perhaps one of the most interesting ones is the concept of an Enterprise Service Bus (ESB) — an example of what Gartner Group calls an enterprise nervous system (ENS) backbone.

The emergence of the Enterprise Service Bus

This is the melding of SOA principles with EAI technology, taking advantage of developments in the broader standards area too, to produce an enterprise integration backbone throughout the value chain with services socketed onto it as required. These services can be running in any environment (J2EE, .NET, legacy application, packaged application, etc.), either within the enterprise or in a partner company. Connectivity to these environments, presen-

tation of data structures and navigation between the services are controlled by the EAI technology.

Figure 3.3 shows the ESB concept. Essentially, an ESB should deliver a powerful, affordable, standards-based backbone that can be applied throughout the enterprise and partner companies. This smoothes the operational path of the processes running the business and reduces the time, effort and cost of integrating the different components that underpin these process steps.

One of the benefits that this approach can deliver is that it allows in-house development teams to build new applications that are already 'integration enabled' and can easily be socketed into the ESB as required. Not only does this save a lot of investment on expensive skills, but it also has a significant impact on time-to-market for new initiatives.

Future EAI standards evolution and attractions

The EAI market has gradually swung from being one driven and controlled by EAI vendors to one where end users and prospective EAI buyers are far more in control. This often happens in markets once they have moved from serving visionaries and leading edge customers to serving the majority of buyers.

Standards have established a strong grip in the minds of today's EAI buyers, in the sense of technical standards such as XML and JMS and also the standard EAI models such as SOAs and ESBs. Standards address all four primary concerns.

In the case of RoI, the emergence of standards forces prices down, while at the same time removing cost from the RoI equation through less expensive skills and reduced requirements for custom code. Standard EAI models, such as the ESB, enable companies to build applications that are already 'integration-enabled' using in-house skills. Also standards-based offerings can be quicker to get going, usually offering more 'out-of-the-box' functionality since there is generally less functional complexity. All these factors improve the overall return and the speed of payback in the RoI calculation.

Incorporation of legacy systems forms one of the largest components of any EAI project, with specialized adapters either having to be built or bought and then tailored to match existing system needs. Both the J2EE Connector Architecture and, more recently, Web Services promise the potential to make dramatic changes to this situation. As JCA takes off, for example, the pressure will be on application package vendors to create JCA interfaces to their products, making it easier to integrate these packages into new process flows. As far as Web Services go, it is unclear at this time exactly how much work is likely to be saved by being able to 'wrapper' a legacy application as a Web Service. Potentially it should reduce the time, effort and business risk of legacy system integration.

The adoption of standards has a simplifying effect on skills requirements. Instead of the market needing pools of people skilled in all the various proprietary implementations of EAI, it tends to reduce this need to one single pool of people skilled in the standards. For example, knowing how to use a specific message server becomes a question of simply understanding how to use any JMS-based server. The result is that since less specialized skills are required, the overall skills requirements are reduced and the skills that are required are likely to be more affordable.

Finally, and perhaps most importantly, the adoption of standards has the potential to reduce business risk considerably. There are several reasons for this:

- **the reduction in need for custom-built code reduces the risk of error and also brings lower maintenance costs and hence total cost of ownership**
- **the reduction in requirements for rare skills reduces the exposure of these employees leaving the company**
- **since standards implement some level of design in exactly the same way in all products, then the areas of design covered by the standards are to some extent 'burned in' and hence are likely to be of improved quality**
- **the flexibility of vendor choice and the ability to change vendors without completely disrupting implementations ensures that the risk associated with vendor lock-in is reduced considerably.**

The long-term value and presence of standards has been reinforced over the last few years by the success of the new, standards-based EAI vendors, both in terms of company performance (often showing growth rates in excess of 100% while other EAI vendors are experiencing falling revenues) and successful deployments as well as by the increasing seriousness with which some of the market leaders are adopting the standards.

Most of the proprietary EAI vendors were quick to include support for the new standards in their offerings, but often the implementation was simply cosmetic, with perfor-

mance and usability issues. There are definite signs that some of these vendors are now making efforts to deal seriously with the standards, either by significantly rewriting their implementations or by acquiring standards-based solutions. This latter course of action can just increase the problem if it results in the vendor having two divergent solution strategies — the key is how well the vendor then converges the technologies.

So it seems clear that standards in the EAI space are truly here to stay. They raise the appeal of EAI solutions while at the same time making it more likely that EAI implementations will prove to be sound business investments.

Choosing winners and losers

Since standards do seem to be here to stay in the EAI market, causing all of the upset and activity discussed in the previous sections, then what does this mean in terms of winners and losers within the market? More particularly, for a company either implementing EAI for the first time or looking to make additional investments in EAI for new projects, how does all this standards activity affect the buying decision?

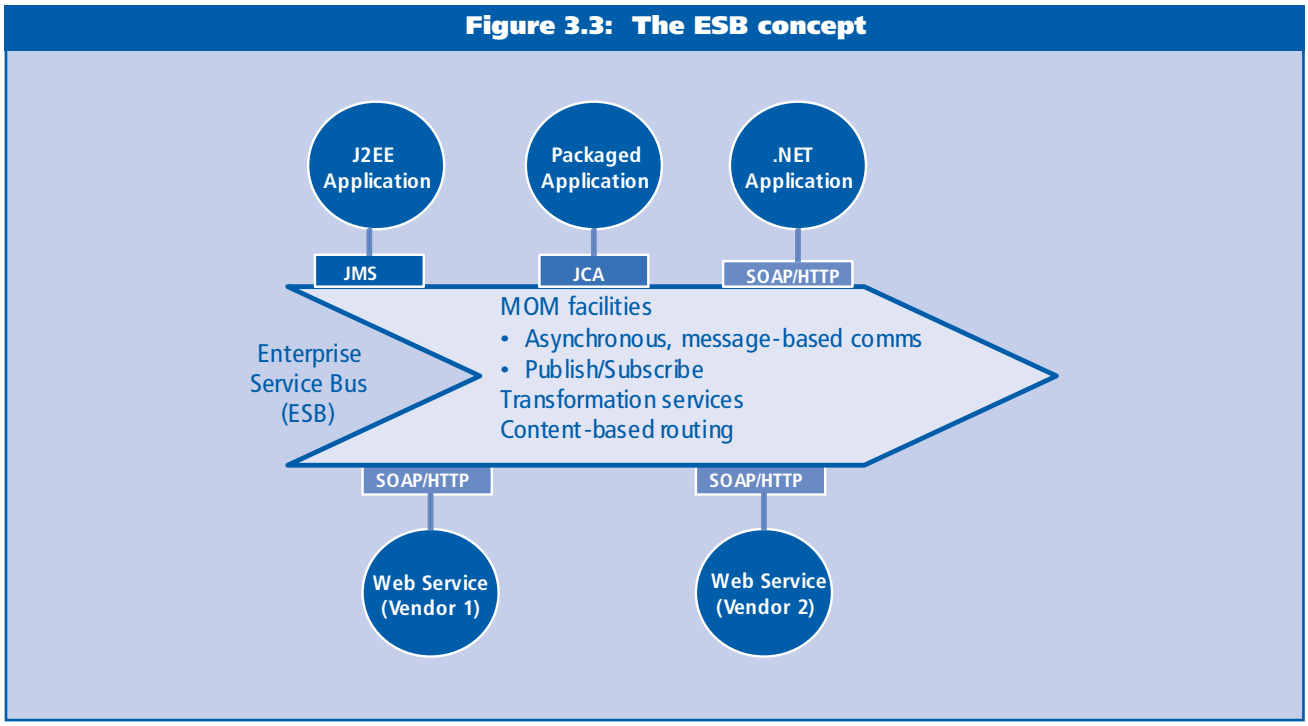
It may seem on the surface that the answer for the organization already using a particular proprietary vendor and looking to expand that usage is obvious, namely that the same vendor should be used. While this could well be true if the expansion in usage is to provide more capacity for an existing project, it may not be the case if a new project is being implemented. Almost all EAI vendors have some level of interoperability with each other, so the projects can be linked together even if different technologies are used.

For instance, it is not uncommon for companies to use one vendor for mainframe-oriented EAI projects but to use another for Web-centric ones. However the skills question is liable to be relevant here. If an organization already has a large pool of skills in the proprietary technology then it may be more advantageous to stay with this rather than having to bring in a new (even if cheaper) pool of talent.

The real issue comes down to how well vendors satisfy the business needs and how well the level of RoI balances the risk. It may be, for instance, that organizations are prepared to pay a considerably higher price for mission-critical software that can be trusted with the most essential parts of business operations, whereas collaboration-style solutions with less stringent service requirements might be much more price sensitive.

The challenges for the proprietary EAI vendors (and those with the biggest share of the market today) are to find ways to embrace the standards as efficiently as possible while ensuring that coexistence with the proprietary implementation is easy. This may be painful for some of these vendors. For example, the most common pricing

Figure 3.3: The ESB concept



model employed is to charge relatively little for the basic server-type EAI functionality but to charge heavily for adapters for different packages and environments. Providing quality support for the J2EE Connector Architecture and Web Services and taking an active role in driving the standards forward threatens a major part of that revenue stream.

Also, since the standards are evolving — and are generally less functionally complete than proprietary implementations — providing an optimal implementation of the standards while still maintaining smooth coexistence with the existing solution can be a real headache.

Then there are the application server vendors that implement some level of EAI functionality internally. In the end the use of application servers will remain reliant on partnerships with other EAI vendors for practical solutions. In general this will favor the standards-based EAI vendors, and in particular the ones who have adopted an SOA approach such as delivering an ESB.

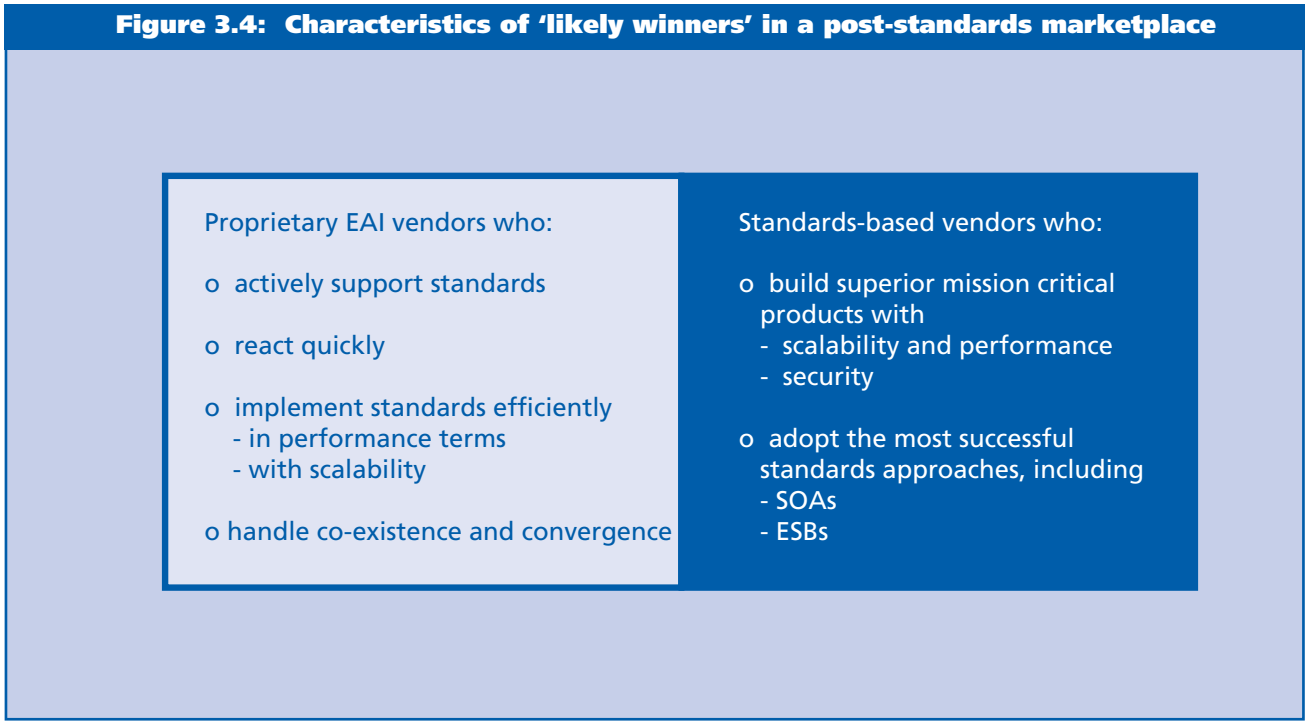
There are two main reasons behind this claim. The first is that application server vendors' prime area of interest is, naturally, in selling more copies of their application server platform and related tools. Their focus is therefore on ensuring that integration within the vendor's own offering works. In fact, offering the customer integration with another platform is seen as a 'Bad Thing'. As a result it is frequently the case that integration with other environments (and especially with other application server platforms) is poor.

For example, it is often the case that, due to differences in implementation, a Web Service under one application server vendor's control cannot be invoked by another vendor's platform. It is this sort of problem that an ESB addresses effectively.

The second reason is that EAI is a complex area to implement well and application server vendors will never be particularly focused on this area because they do not see it as an area of core competitive competence.

For the standards-based EAI vendors, although they have the luxury of having already adopted the standards with all the benefits discussed previously, the challenges are different. The winners will be those who can differentiate themselves through their implementations, for example providing production-quality values such as performance, scalability and security while at the same time remaining adaptable and retaining their superior ease of use and RoI. Figure 3.4 summarizes the characteristics of likely winners, and points of consideration to use when selecting an EAI solution — whether for the first time or for a new project.

Figure 3.4: Characteristics of 'likely winners' in a post-standards marketplace



Management conclusion

The EAI market has redefined itself since its inception to encompass the functionality necessary to support business integration. EAI includes areas such as:

- ***messaging***
- ***application integration***
- ***process integration***
- ***work flow.***

Power has gradually shifted from vendors to buyers and implementers. This simple fact, combined with the importance of integration as a business priority and the inherent complexity of EAI technology as a whole, has made the EAI market fertile ground for the development of industry standards.

In turn, these standards have made the jump from academic exercises in technical purism to important initiatives in raising the value of EAI. Many, as discussed above, have already achieved widespread acceptance and adoption. Standard models for getting the best out of EAI have also emerged. These developments have affected the dynamics of the EAI marketplace:

- ***current market leaders are trying to claim compliance with minimum effort***
- ***other vendors are leaping on the standards bandwagon to seize a share of this highly lucrative market***
- ***end users are enthused at the thought of consistent, standards-based offerings, all competing on a level playing field, forcing down prices and reducing the level of business risk.***

Not all existing EAI vendors will survive this change. Current leaders might not step up to the standards challenge while others will ensure future success. New entrants will live or die based on how well they can establish a real presence and prove their 'big player' credentials. The one certainty, however, is that end user organizations implementing solutions in the new, standards-based EAI world will be the big winners.

Enterprise integration: buses and brokers

Tom Welsh
Consultant

Management introduction

Somewhere along the path from Enterprise Application Integration (EAI) to Business Process Management (BPM), integration brokers and message brokers entered the IT industry's awareness. So did message buses and enterprise integration buses, not to mention Web Services, Service Oriented Architectures (SOAs), choreography and orchestration. All in all, there are a lot of acronyms, initials and jargon terms to understand.

If the IT industry is going through a confusing period, with many alternative approaches to enterprise integration being promoted, the underlying problem itself is easily grasped. Organizations have invested in large amounts of computer equipment, software applications and networks. How can they maximize their RoI by making the various applications to work together?

Many products on the market promise to solve the problem of integration, but users will always remain responsible for choosing and deploying the right technology to further their business goals. As Tom Welsh argues, the secret — inasmuch as there is one — lies in:

- ***accepting that responsibility***
- ***never believing that it can be delegated to software, however apparently 'intelligent'.***

Enterprise integration

The EAI market was born in the late 1990s. For years it had been known that, for various reasons, an organization's departments, offices and divisions tended to commission their own applications. As a result, a company could have hundreds of business applications and databases, each of them vital for its successful operation. But too few of these could communicate with any of the others. As a way of illustrating this isolation, such applications were dubbed 'stovepipes' or 'islands of integration'. Both these terms were commonplace by 1990.

In his authoritative book 'Enterprise Application Integration', David Linthicum defines EAI as "the unrestricted sharing of data and business processes among any connected applications and data sources in the enterprise". That is all very well and good, but it only takes us so far. What exactly are the applications to do with these data and business processes once they get hold of them?

The advocates of EAI and BPM talk excitedly about the prospect of building a 'common virtual system' that links together all your IT assets. Fine, but how many of those systems were even designed to deal with related subjects? There is often little point in trying to get your Accounts Receivable application talking to your manufacturing system, or integrating your CRM with your R&D labs.

Furthermore, even if two applications do inhabit the same universe of discourse, are their data schemas compatible? If one application and its database think 'client' while another thinks 'customer', it is going to take much work to bridge that seemingly insignificant gap. In general terms, schema translation is still a research topic, although the W3C's Resource Definition Framework (RDF) is a promising start.

In an ideal world, every organization would have a complete and consistent enterprise metadata repository, to which all applications would conform. But in reality, this is about as easy to find as an organization that has no application stovepipes in the first place.

The executive impulse

By the late 1990s, the necessary conditions for EAI were falling into place. Processors, storage and particularly networks were becoming much smaller, cheaper and faster. Many processing and communication functions that had previously been impossible or unaffordable were coming into consideration. A huge amount of R&D had gone into multi-tier client/server — one offshoot of which was the application server.

Last but not least, awareness of corporate stovepipes and islands of integration had spread upwards to CxOs and the boardroom. At a time when spending on IT was spiralling ever upwards, astute sales people found that they could readily interest top management in the concept of doing more with less.

The sales people were happy, as this gave them an opportunity to achieve their ideal situation — 'selling high' to executives with spending power and the authority to make important decisions quickly, and make them stick. These executives, too, were happy at the prospect of getting their huge (and previous incompatible) IT resources to play efficiently together.

How was it all to be done? The basic ideas were simple enough, requiring nothing beyond the current state of the practice. Given ***n*** computers, each running an important application (for simplicity's sake), you could connect each to all the others. That would yield ***n2 — n*** links, or, allowing for bi-directional links, ***(n2 — n)/2***. That is 6 links for 4 computers, 10 links for 5 computers... and 190 links for 20 computers. Rather unmanageable. These are, of course, not physical (or even logical) network links, but unique program-to-program software interfaces.

Using a central hub, however, the number of links needed is just ***n***. Each computer has a connection to the hub, and that is all. This is a far simpler topology — which is why, for instance, many airlines prefer (or preferred) a hub and spoke layout for their services.

Behind the apparent simplicity of the hub and spoke geometry, though, lurk further ramifications. Most obviously, the hub becomes a potential single point of failure, so if the overall system is to deliver any kind of guaranteed up-time, the hub must be made fault-tolerant.

More to the point, the number of software interfaces does not necessarily bear any relation to the total number of applications or computers. At the other extreme, they might all be running a co-ordinated suite of products from a single vendor — making the purchase of an EAI product expensive overkill.

Adapters

What really matters is how many of an organization's applications — those that ideally should communicate with one another — are actually unable to do so. For instance, a SAP ERP system might not be able to access CICS directly. To make this possible, both could be linked to an EAI hub which would need to be able to translate SAP requests to

CICS. That would require the creation of two adapters, one for SAP R/3 (say) and one for CICS. In many cases, these must be able to perform transactions and maintain security requirements (which significantly complicate the development task).

Further problems are posed by the multiple levels of abstraction that are commonly lumped together under the heading of 'semantics'. When programmers talk about semantics, it is usually in connection with how their software should interpret a given piece of data. Is it an address, a value or, perhaps, a descriptor? Adapters take care of all these technical details, but it is harder to incorporate an understanding of what the given fields mean in terms of the overall flow of processing. This is part of a very important and fundamental issue — that of shared vocabularies or ontologies — with which the BPM and Web Services communities are only now coming to grips.

Intuitively, there are two ways of setting out to write an adapter. The hard way is to list all the data elements and functions of each application, and create mappings for every single one of them. That could take years.

Fortunately there is an attractive shortcut. This is to look at the applications to be linked, and implement only those data elements and functions that are currently in use. Unfortunately, such a minimal mapping is liable to be broken every time the calling application is modified in such a way that it needs to invoke the called application in a new and different way. Some working compromise must be sought, but this is not an easy choice.

Even in the case of a comprehensive mapping, the adapter writer's work is never done. Every few years, major new versions of applications are released, which may require extensive adapter rewrites. This problem has been elegantly addressed, for instance, by Intalio with its Intalio|n3 BPMS (Business Process Management System). By enabling users of ERP products (like SAP's R/3) to define and store customizations outside the ERP system, Intalio promises them a double win:

- **first, the customizations are much easier to create in the first place**
- **second, they persist through updates of the ERP system, saving the enormous effort that would be required to re-create them from scratch.**

Buses and brokers

Buses and brokers are the two fundamental components of any EAI system. For our purposes, we can regard BPM and other, more elaborate architectures as being layered on top of an EAI infrastructure.

At the bottom, we need physical connectivity: this is now ubiquitous in the shape of TCP/IP, whether over WANs or LANs. Then we need synchronous or asynchronous middleware (or both). This is provided by:

- **RPC**
- **RMI**
- **DCOM**
- **CORBA**
- **message-oriented middleware (MOM)**
- **Web Services and .NET Remoting (the most recent developments).**

Some forms of MOM are generally deemed to be message buses or enterprise integration buses in their own right. For instance, the Java Message Service (JMS) is widely referred to as an Enterprise Integration Bus, in view of its dual role as a standardized part of J2EE and an interface to powerful, mature MOM products like IBM's WebSphere MQSeries Enterprise. Broadly speaking, an enterprise message bus is any form of middleware which can be deployed throughout (and between) organizations and that provides:

- **queue management**
- **guaranteed delivery**
- **determinable qualities of service (QoS).**

Yet the whole concept of an enterprise integration bus is slightly suspect. A bus, after all, is defined as a communications channel that connects all nodes directly together, without the overhead of going point to point.

At the physical level, however, such a creature is close to non-existent. All TCP/IP networks function by handing packets around, and a typical link requires half a dozen 'jumps' through intermediate nodes. At best, when people talk about an integration bus they mean a network that gets messages through reliably, with a minimum of delay, to all participating computers.

Moving on to the more interesting topic of integration brokers, one can find a remarkable variety of differing products lumped under the same broad heading. This is partly because the technology has outstripped standardization. As there is no industry standard integration broker, nobody can be entirely sure what one looks like.

The ideal integration broker (sometimes called a message broker) combines:

- **the roles of EAI hub, schema translator and business rules engine**
- **with other desirable functions such as security, remote management and transaction auditing.**

It may also incorporate a metadata repository. Most integration brokers offer a wide variety of adapters for popular applications servers, middleware, databases and packaged applications. Unfortunately, most present-day integration brokers are proprietary, which means in practice that the more elaborate their features, the more powerful is the vendor lock-in effect.

There is certainly no shortage of contenders. SAP, Siebel, PeopleSoft and other packaged application vendors all have their own integration brokers — naturally, as they all have extremely complex and extensive schemas, parts of which occasionally have to be translated to interoperate with external applications. Actional (previously Mercator), IBM, Microsoft, TIBCO and webMethods are other important players.

Microsoft's BizTalk Server may have a big future here, not least because it offers a lower price than many competitive integration brokers. On the other hand, it also leaves rather more of the work to the user. BizTalk Server is a significant product, not only because it is Microsoft's entry in the EAI stakes but because it is a halfway house between the proprietary EAI of the 1990s and the coming wave of 'universal' integration based on XML and Web Services.

Uniquely, BizTalk Server runs on Windows only. But this is not really a handicap, because it connects with other nodes through Web Services (among other protocols). So a BizTalk Server running on Windows 2003 Server, for instance, can be dropped into an all-UNIX IT environment, and still function smoothly with a minimum of tuning. In that sense, this makes BizTalk the best example to date of an integration broker that is independent of any specific integration buses.

Web Services and EAI

Many vendors, and some analysts, believe that Web Services will decisively affect the enterprise integration business by simplifying and speeding up development, pushing price points lower and thus turning integration into a mass market. There certainly are some fat margins to be attacked among the current sector leaders. Annrai O'Toole, CEO of the Dublin-based Web Services specialist start-up Cape Clear, is one of the most vocal and articulate advocates of this view.

"Web Services is *(sic)* going to succeed where EAI has failed," he predicted last year, "by offering a fast, easy and affordable means to tie together applications and systems, all based on open industry standards. We'd like to welcome everyone to prêt-a-porter integration."

O'Toole feels that the enterprise application integration (EAI) market is ripe for the picking as incumbent vendors such as webMethods, SeeBeyond, TIBCO and Vitria have priced their products far too high, and consequently limited their market penetration. As a typical example, he cited TIBCO, whose BusinessWorks — a product ostensibly aimed at small to medium enterprises — was launched at the remarkable price of $450,000 per server.

According to O'Toole, Web Services embody hardly any new ideas but nevertheless change the whole economics of integration. "Over time, there won't be any such thing as connectors to SAP and other applications," he explained. "Instead, these products will be exposed as Web Services."

Cape Clear's CapeConnect automatically exposes Java, .NET and CORBA components as Web Services without any additional code being written. It can be run either standalone or as an application within a J2EE application server. It supports EAI through mapping software which can connect any XML data source to any Web Service. Business process logic can be inserted into the message stream through an Interceptor API, and a range of APIs are exposed — including:

- **a SOAP parser**
- **an authentication module**
- **serialization**
- **transports**
- **SOAP headers**
- **HTTP Basic Authentication.**

It is only prudent to point out that, while Web Services can apparently undercut proprietary integration products quite substantially in terms of price, they have a lot less functionality to offer.

This may be a good bargain at the low end of the market, where requirements are less demanding. But, for enterprise applications, there is still no credible Web Services offering that can offer a challenge to the well-established EAI solutions sold by the likes of IBM, TIBCO, webMethods, SeeBeyond and the rest. The 'plumbing' is more or less ready, but advanced features — like transactions, security, queuing, buffering, data transformation and BPM — are still mostly on the Web Services drawing board.

Nevertheless, energetic young companies — for example, Cape Clear, Intalio, PolarLake and Vitria — are working hard to provide complete enterprise integration and BPM solutions on top of the standard Web Services stack. When they have done so the older generation of EAI and BPM products may look both outdated as well as overpriced.

The emergence of standards

The Web Services stack has burgeoned since IBM, Microsoft and their partners started the process by submitting SOAP 1.1 to W3C back in May 2000. Admittedly, specifications have been written and submitted faster than standards consortia have been willing to adopt them formally — but this inconvenient time-lag may be less noticeable in future, as IBM and Microsoft have shifted the focus of their work from W3C to the more flexible OASIS. So far, only SOAP 1.2 and UDDI 1.0 have been promoted to the status of industry standards. Within a year or two another dozen or so specifications will probably have followed suit.

SOAP, WSDL and UDDI are the three legs on which Web Services stand:

- **SOAP is the indispensable application protocol**
- **WSDL the service description**
- **UDDI the directory.**

Far from being cut off from all previous experience, the essential Web Services standards are themselves open. For instance, SOAP is usually implemented on top of HTTP, but it can now also run over SMTP or even JMS (the Java Messaging Service).

The last-named, in turn, is a good example for it is no more than an interface specification. This means that it can be implemented by anybody — which is why most of the leading MOM products have done so as evidenced by WebSphere MQSeries Enterprise, Sonic MQ and SpiritWave.

These provide high levels of reliability as well as advanced features like controllable quality of service (QoS) or 'once-and-once-only' delivery. Provided users stick to the standard SOAP and JMS interfaces, they retain the option of switching the underlying product with (relative) speed and ease.

The WS-Coordination and WS-Transaction standards (when agreed) will take care of transactions — including the long-lasting ones that can be expected to occur in the course of e-commerce. WS-Security and the flock of ten other specifications that follow it, like ducklings behind their mother, deal with all the many and various issues of security, integrity and privacy that will need to be addressed before the commercial world has confidence in enterprise Web Services.

Last, but not least, BPEL4WS provides a powerful, flexible language for sequencing Web Service requests and responses — an activity that has come to be known as 'orchestration' or 'choreography'. These terms are not quite equivalent:

- **choreography is generally interpreted as being more abstract, declarative and top-down**
- **orchestration is more detailed and procedural.**

BPEL4WS derives from the merger of two earlier languages, Microsoft's XLANG and IBM's WSFL. According to Intalio, it owes much to BPML, which was explicitly designed to combine the strengths of the Petri Net and Pi Calculus approaches to BPM. After initial skirmishes, it looks as if the whole industry is going to settle down in support of BPEL4WS, which is promoted by IBM and Microsoft and, in any case, contains an adequate set of BPM features.

Management conclusion

Enterprise integration in its fullest sense is an inordinately difficult and demanding task. Today there are two primary categories of enterprise integration solution, those that:

- ***have proved their worth and are widely deployed***
- ***are entirely based on industry standards.***

Unfortunately, the former are mostly proprietary — which means they differ in many ways, and are not interoperable. In contrast, the latter are not yet ready for use.

There is a good reason for this state of affairs: namely, it is much easier to talk about enterprise integration than to accomplish it. The ad-hoc proprietary solutions that sprang up first go some way toward solving the problems of integration, but they are limited, expensive and — worst of all — generally not interchangeable. We can think of them as prototypes for the industry-standard BPM systems to come.

The Web Services and BPM communities are currently engaged in a Herculean effort to agree and implement standards for enterprise integration and, yet more ambitiously, for integration between enterprises. It will be some

time before these are complete, and a good deal longer before standards-compliant implementations are available. Then there will be a period of learning, during which the standards are edited to avoid the shortcomings that will inevitably materialize.

The obstacles to be overcome are of two fundamentally different kinds. Ironing out the wrinkles in today's standards, and finding the best ways of re-using the technology that is already understood, will take a few years. But we can be sure that these things will be done.

The big problems are in the area of schema translation and methods for agreeing shared vocabularies. There is no guarantee that those problems even have general solutions at all.

Are we ready for transactional Grids?

Mark Lillycrop
Principal Analyst
Arcati

Management introduction

The scientific credentials of Grid Computing are now well established. Grids have been, and are being, designed to harness vast amounts of compute power, using common open standards, to tackle vast problems in a fraction of the time that more conventional IT solutions allowed. Take a look at some of the research projects described on the www.globus.org *Web site and you will see where Grids really come into their own:*

- ***from collaborative medical applications***
- ***to space exploration***
- ***to petrochemical analysis.***

Grids offer endless opportunities to crunch data in ways that were unthinkable before.

But ever since Grid applications started to appear in academia, commercial IT vendors have been looking for ways to extend the technology to the wider business arena as a way of providing vast, flexible computing resources that could be supplied 'on-demand'. Yet, despite all the talk, such applications remain elusive. In this analysis, Mark Lillycrop examines what progress has been made, and by whom, in the search for transactional Grids.

Progress to date

For years the talk has been about the need to harness redundant PC MIPS during the long hours when machines lie dormant on desktops. Microsoft has experimented with ways to take advantage of spare PC disk space over a network. It really should not require a giant leap of imagination to find a way to run overnight batch jobs in this way, harnessing PC processors as and when required.

So far, though, the only way that PCs have been Grid-enabled is in such projects as the extra-terrestrial SETI initiative. In the business world, we still switch off our PCs at 5pm and turn them on again in the morning — or, worse still from an environmental viewpoint, leave them running but doing little.

Nevertheless, there is huge interest in exploiting Grid technologies within the commercial environment. For companies such as IBM, HP, Sun and Oracle, Grid computing is now a key part of the strategic marketing message, as they work to provide support within their 'Utility Computing' offerings. Over the next few months, for example, we are likely to see interesting developments in database design, with suppliers adapting existing data structures to the specific requirements of Grid applications.

IBM is perhaps the vendor that is doing most to raise the level of acceptance of Grid computing outside the scientific and academic communities. Its Grid 'Focus Areas', published in January 2003, made an admirable attempt to:

- **express Grid solutions in terms of specific business problems**
- **define a number of vertical markets and horizontal functions**
- **provide decision trees that could enable customers to choose between various Grid tools.**

These 'Focus Areas' demonstrated that Grids were applicable to financial services and 'government development', as well as the more scientifically-oriented aerospace/automotive and petrochemical sectors. Figure 5.1 shows one such sample Focus Area.

But, despite IBM's claims that 'enterprise optimization' takes Grids into server consolidation and capacity-on-demand territory, commercially-oriented Grid applications remain restricted to number crunching such as found in:

- **long-term financial forecasts**
- **demographic analysis**
- **complex business intelligence systems.**

Grid is a very broad term, and some so-called Grid offerings are arguably little more than clustered systems. They apply much the same principles of allocating processing tasks to engines on the fly, but on a much smaller (and more tightly managed) scale than mainstream Grids. For all the positioning and statements of direction, Grid technology is largely locked in the technical computing (scientific engineering) world where it was born.

Redefining what a Grid is

That said, it is interesting to consider how IBM actually views the Grid market. The official 'Blue' definition of Grid Computing used to be: "Distributed computing over a network, using open standards to enable heterogeneous operations".

Compare that with the definition that IBM now prefers: "Grid computing enables the virtualization of distributed computing and data resources such as processing, network bandwidth and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities".

Instead of viewing the Grid concept simply in terms of distributed processing, the focus has now broadened to cover the data, storage and network components of a 'virtualized' computing environment. The implication here is that Grids will ultimately offer a vast pool of IT resource into which users will be able to tap — the ultimate goal of 'on-demand' Utility Computing.

Before we can hope to reach that goal, though, many issues need to be addressed. We already know how to share processing over a Grid. A huge amount of work is currently underway to improve the management of data and storage. But several obstacles still stand in the way of commercial realization:

- **security is still uncomfortably basic**
- **service levels are unpredictable (because nobody has overall control of the participating computers)**
- **few applications exist outside the scientific environment.**

We cannot begin to think of Grid computing as a satisfactory answer to general-purpose Utility Computing until these issues are addressed. In particular, we need to work out exactly how transactions and service levels are going to be managed within a Grid world.

Grids and Web services

Grids and transactions are not (yet) comfortable partners. According to one of the UK's leading proponents of Grid technology, Dr Mark Parsons of the University of Edinburgh Parallel Computing Centre, it is still early days for Grid computing within the transaction management environment. "I hear people say that Grids are old news now," he said, "but the work has only just begun. It could be ten or fifteen years before we see real commercial progress. In the scientific world, there was a strong need for Grid-based solutions, and we knew what we were trying to achieve. Universities were used to co-operating and sharing resources, and they understood how they could benefit from harnessing and sharing computing power on an ad hoc basis."

He continued: "financial and commercial businesses running transaction-oriented applications have different priorities, and are reluctant (for security and competitive reasons) to co-operate in the same way as academic institutions. We have yet to see how Web Services and Grid services will come together to provide an on-demand computing environment for commercial companies, but it will happen in due course. Many leading IT companies have staked their future on this."

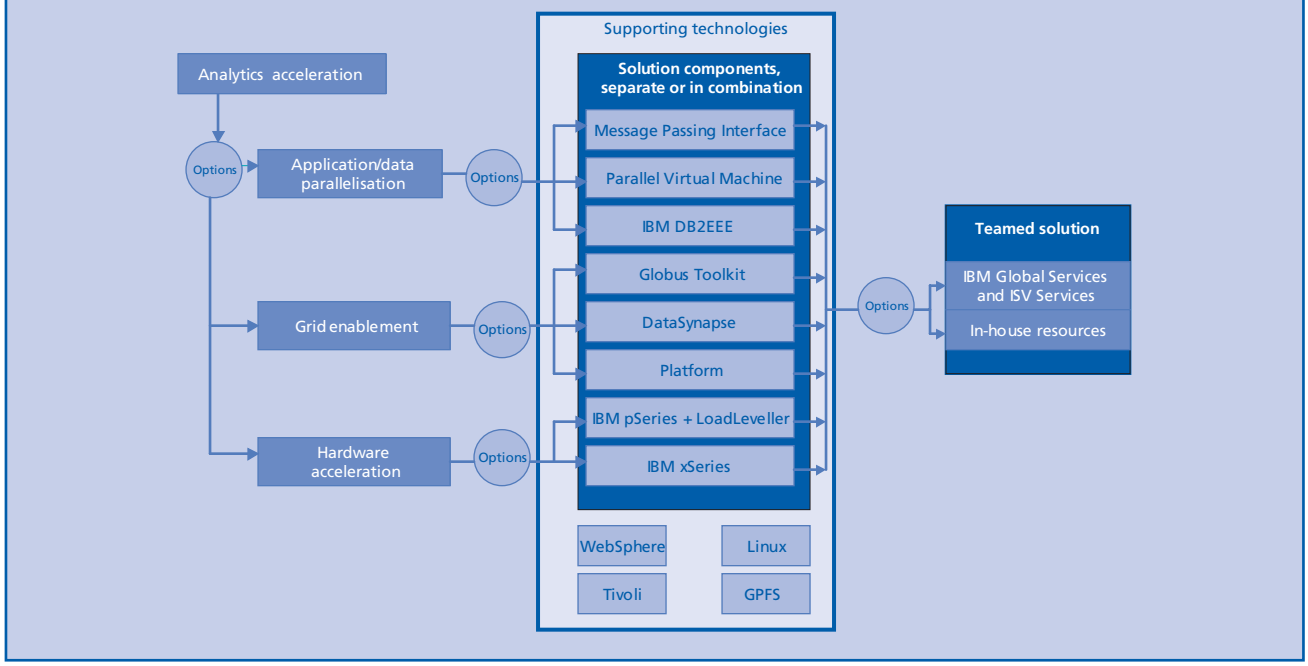
The important issue for the proponents of Grid technology is to ensure that we do not re-invent the wheel. While Grids are not yet well prepared for handling transactional workloads, a number of research efforts are in place to address the issue. At the same time, however, similar developments are underway in the Web Services arena. The two communities have similar and overlapping objectives and, as Dr Parsons suggests, it is to be hoped that the two efforts will evolve and converge in ways that will be determined — or shaped — by commercial customer requirements.

Perhaps the single most significant initiative in this respect is the Open Grid Services Architecture (OGSA), an industry-wide standard which is evolving under the auspices of the Global Grid Forum. In its draft document, Physiology of the Grid2, the OGSA architects explain their thinking. "We assert that, regardless of their differences, developers of applications for 'virtual organizations' face common requirements as they seek to deliver Quality of Service — whether measured in terms of common security semantics, distributed work flow and resource management, coordinated fail-over, problem determination services or other metrics — across a collection of resources with heterogeneous and often dynamic characteristics."

The OGSA is complex and still evolving. In essence it:

- **takes the best parts from the Grid community and from Web Services**

Figure 5.1: An example of a 'Grid Focus Area'



- **builds on the standards and protocols embedded in the Globus development toolkit.**

The Globus Toolkit itself is an Open Source set of services and software libraries, designed to aid the rapid development of standard Grid applications. But it is indisputably scientifically oriented. Its key components — such as Grid Resource Allocation and Management (GRAM), the Meta Directory Service (MDS), and the Grid Security Infrastructure (GSI) — are all intended for compute-intensive applications, with little appreciation of the commercial world.

Ironically, GRAM, MDS, and GSI address much the same issues as equivalent commercial standards:

- **managing the remote completion of a task**
- **monitoring system and network status**
- **controlling authorization as well as other security attributes.**

However, these facilities are rudimentary in comparison with commercial transaction management expectations. The reality is that they are intended to address the Grid's 'transient service instances', rather than the 'persistent services' required in many business applications.

The OGSA basically brings together the Grid standards with the evolving protocols of the Web Services environment: SOAP, WSDL, WS-I, and UDDI. WSDL (Web Services Description Language) is at the heart of this integration effort, as it is increasingly being used to define services and interfaces in both Web Service-based as well as Grid applications.

By leveraging the combined strengths of each environment, it is hoped, the OGSA will:

- **provide a way to develop system, database, and work flow tools for the Grid environment**
- **introduce the time-critical service functions that are essential to managing commercial transactions.**

At the same time, Web Services will likely be extended to support some of the transient service characteristics of compute-intensive scientific application.

As Parsons points out, these developments are taking place slowly, and require not just industry-wide support but considerable cultural change. With so many IT vendors now relying on Grid and Web Services for their future direction, we are likely to see standards being pulled in two directions, particularly between:

- **the Microsoft .NET supporters**
- **the proponents of J2EE, where much of the initial development has taken place.**

Inevitably, though, we will see common services emerge that blur the distinction between Grid and Web Services. At the same time, business-critical transaction support will become increasingly viable.

Grid versus Hive

In view of the ponderous nature of these developments, and the natural impatience of the IT industry, it should not be surprising that some vendors are keen to tell us that the Globus/OGSA route is not the only alternative. Some argue that we will never succeed in providing transaction management within a Grid-type environment unless we reconsider exactly what we are trying to achieve.

Chris O'Leary, VP of Marketing at Tsunami Research, puts it this way: "To a degree, it depends on how you define Grid computing. One sense is the specific work with regard to OGSA and the Globus toolkit. A second sense is what is more generally known as distributed supercomputing or utility computing. Most of the work to date has been directed at working out methods that will deliver the basic functionality of a Grid. To use the power grid analogy, most of the work has focused on issues like building the power transmission lines and the billing systems.

"Having said that," he went on, "we think the opportunity exists to build transaction processing systems that leverage many of the benefits of the power grid model. For example, many organizations have seasonal spikes in demand. The problem is that organizations have to size their systems for that week or two of peak load but spend the rest of the year running at only 50% or even 10% of the peak.

"What if you could create a Grid-based transactional resource that would enable organizations to borrow transaction processing power to handle those peak times rather than having to buy it all themselves? Of course, to do transaction processing, such a system would have to possess a sense of time (specifically, to know how long a transaction has been running). That is something the existing companies in the space do not address."

Tsunami's solution to the problem is Hive Computing, an approach which displays many of the characteristics of Grid Computing but which allows transactional service delivery issues to be addressed. The Hive approach aims to provide reliable transaction processing, using:

- **low-cost dedicated commodity computers**
- **Grid principles to allocate processors to tasks**
- **the notion that hardware is fundamentally prone to failure, and achieves guaranteed reliability by removing the single point of failure by using a Grid approach to allocating tasks to processors.**

A Hive is a virtual collection of computers, or blades, known as 'workers', which are all:

- **connected to the same logical network, though not necessarily physically co-located**
- **dedicated to the Hive**
- **running the same software, regardless of hardware configuration.**

The Hive receives and processes requests sent to it by a 'client' — which can be a standalone application, a PoS terminal or an application server (Figure 5.2). The Hive allocates the request (typically a piece of data with an associated service) to a worker, which then responds to the request and returns the completed task and data to the client. The Hive:

- **monitors the worker throughout**
- **makes sure that the request is handled within a satisfactory time**
- **switches in other workers if problems occur.**

In some ways, the Hive does not look much like a Grid at all, particularly in its use of tightly managed, homogenous, dedicated blade PCs. On the other hand, it:

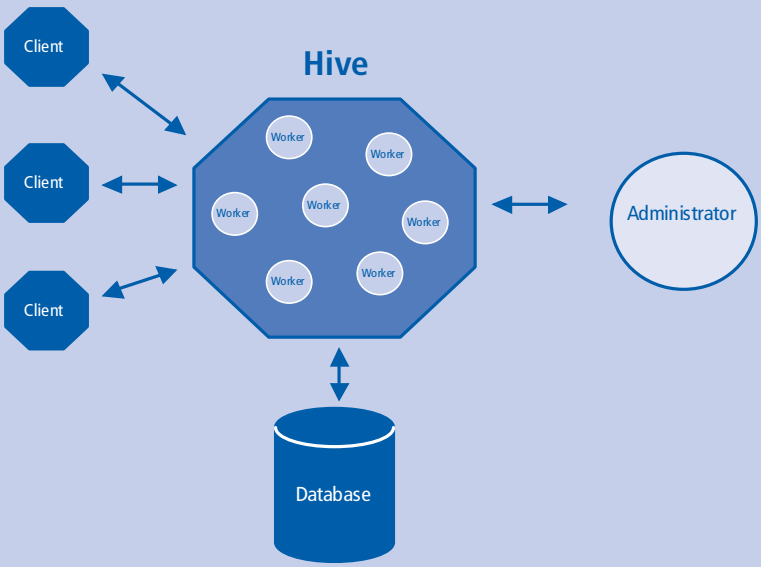
- **offers a highly scalable architecture (anything from tens to thousands of computers can be plugged into a single Hive, and several Hives can be connected together as a Transactional Grid)**
- **displays many of the self-healing, self-maintaining traits that IBM and others are beginning to offer with their Grid solutions.**

The Hive is not designed for the serious number-crunching of traditional Grids, and it does require some specific modification of the application. Furthermore it also provides (built around commodity hardware) what the Grid does not at the moment:

- **a high level of central control**
- **a clear understanding of response times and service levels.**

Tsunami acknowledges that its solution will not be suitable for every commercial environment or problem, particularly those involving much legacy code. But, as a way of

Figure 5.2: The Hive approach



managing transactions within a Grid-style environment, it is certainly worth watching. In time, this may be the way that mainstream transactional Grids evolve.

Management conclusion

Clearly there is intense interest in the way that Grid computing will develop over the next few years. It is important to keep an eye on support for transactions and commercial activities. Inevitably, Grid and Web services will come together through initiatives such as OGSA, if only because so many of the underlying protocols offer overlapping function: it makes no sense to develop two standards in parallel when the industry's efforts can be focused in one direction.

Having said that, Mr. Lillycrop argues that we also need to guard against excessive vendor hyperbole. If Grids are neither ready nor appropriate for supporting transactional systems, there is no reason why this should occur. In the meantime, without excessive claims, vendors — with their eyes on grand-scale server complexes for Utility Computing — will be able to examine the feasibility of supporting transactions across new types of architecture. This also fits with commercial IT, customers who are remarkably conservative when it comes to the adoption of new standards and technologies.

If academia is comfortable with radical ideas, experimentation, collaboration and slow evolution, business IT is concerned with predictable service levels, rapid development, watertight security and, increasingly, 24x7 availability. Given the similarity of many Grid and Web Service standards, it is equally remarkable how dissimilar the priorities and motivation of their respective users are.

However fast the technologies evolve, real change will take place only when the buying customer is ready for it. This may yet prove to be the single biggest restraint on transactional computing emerging which uses Grid technologies

Service bus and container middleware

Dr Keith Jones
IBM Software Solutions Worldwide

Management introduction

Enterprise IT departments are constantly under pressure to deliver integration projects more quickly and to tighter budgets. This has never been more true than in the current business climate. Diverse platforms, execution environments and programming models have all contributed to a complexity that often defeats the best integration teams.

But help may be on its way as middleware vendors focus on developing and implementing a new generation of infrastructure which is standards-based and focused on the service-oriented approach. By eliminating dependencies between clients and service providers, this service-bus-and-container infrastructure promises to reduce complexity and pave the way for less expensive integration of application systems.

In this analysis, Keith Jones looks for the value that service-bus-and-container technology will deliver as middleware products evolve and emerge. Greater IT flexibility and responsiveness would seem to be the value most often sought after, and promised by, proponents.

Service-oriented integration

As many enterprises struggle to maintain focus and funding on critical integration projects, middleware vendors are quietly developing the next generation of software infrastructure. The full details have not yet been revealed but early implementations of some of the key technologies are already emerging.

The motivation for this next generation of infrastructure comes from three important drivers:

- **legacy enterprise application inventories continue to grow inexorably**
- **systems platform investments continue to diversify**
- **skill and other resource costs continue to escalate.**

And yet to develop and maintain competitive advantage — particularly in an 'on-demand' world — enterprises require ever increasing IT flexibility and responsiveness. In answer to this seemingly impossible requirement, the service-oriented approach has been proposed by industry thought leaders. To satisfy this, and the initial enabling Web Services tools, new technologies are being made available by leading vendors. Much more thought and enabling is needed before the approach is fully realized, however.

The service-oriented approach (Figure 6.1) encourages systems architects and implementers to think about composing strategic new and legacy business functions as services. The premise is that new functions can be built economically by re-using certain services and developing those that do not already exist. When successful, this achieves both improved development productivity — through re-use — plus reduced time-to-market.

Many enterprises claim to have implemented service-oriented systems in the past; some with outstanding success and some with mixed success. The difference between those projects and the current initiative is that:

- **the new approach is explicitly based on open industry standards**
- **it applies to both intra- and inter-enterprise domains**
- **software vendors are in full support, making available a variety of service-oriented tools and the emerging middleware technologies are more likely to be delivered in a reasonable timescale.**

If this direction is maintained, with evolving standards and corresponding inter-operating implementations for diverse systems platforms, enterprises will benefit from:

- **a growing pool of available skills familiar with the relevant standards and technologies**
- **facilities for exposing standard service interfaces for, and re-use of, business functions independent of the platform on which they run**
- **flexibility to offer services both to internal users, and external users — suppliers, business partners and customers — in like manner.**

However, before leaping to the conclusion that the service-oriented approach has completely satisfied the seemingly impossible requirement currently placed on IT departments, we must look more closely at:

- **the proposed approach**
- **how middleware infrastructures are evolving in support this.**

De-coupling to reduce complexity

The essential separation between the service layer (Figure 6.1), containing service interfaces, and the component layer, containing implementations, is just one degree of freedom required to deliver IT flexibility and responsiveness. This is the freedom to modify a service implementation without requiring corresponding changes to clients to ensure continued operability.

Dependencies such as this, between clients and service implementations, are often:

- **the source of complexity in deployed systems**
- **the root cause of IT inflexibility and operating costs.**

Several other degrees of freedom are required to fully decouple clients from services for optimal flexibility and responsiveness (Figure 6.2). Each of these has been addressed in existing or next generation middleware technologies.

By developing standards-based technology and implementing it on a wide range of system platforms, middleware vendors are removing the dependency between clients and services running in a network of heterogeneous nodes. The standards-based approach, if all goes as intended, should remove other dependencies on operating system, execution environment and implementation language technologies.

Yet there is always an assumption that the several platform implementations will inter-operate with each other. Organizations such as WS-I (see *www.ws-i.org*) are working to ensure this.

By publishing standards-based service descriptions the contract between client and services should also be free of any dependency upon implementation technology as noted above. The chosen standard, WSDL (see *www.w3c.org*), facilitates definition of the information model, messages and operations that are the basis of abstract interface contracts between clients and service providers. Other standards will describe the qualities of service provided with usage.

The format of messages exchanged in a service network is defined using the XML standards (see *www.w3c.org*) and the packaging of service request and response information defined by SOAP (see *www.w3c.org*) and other related standards. For example, SOAP defines a header for contextual information such as security signatures and a body for business specific information with attachments as required.

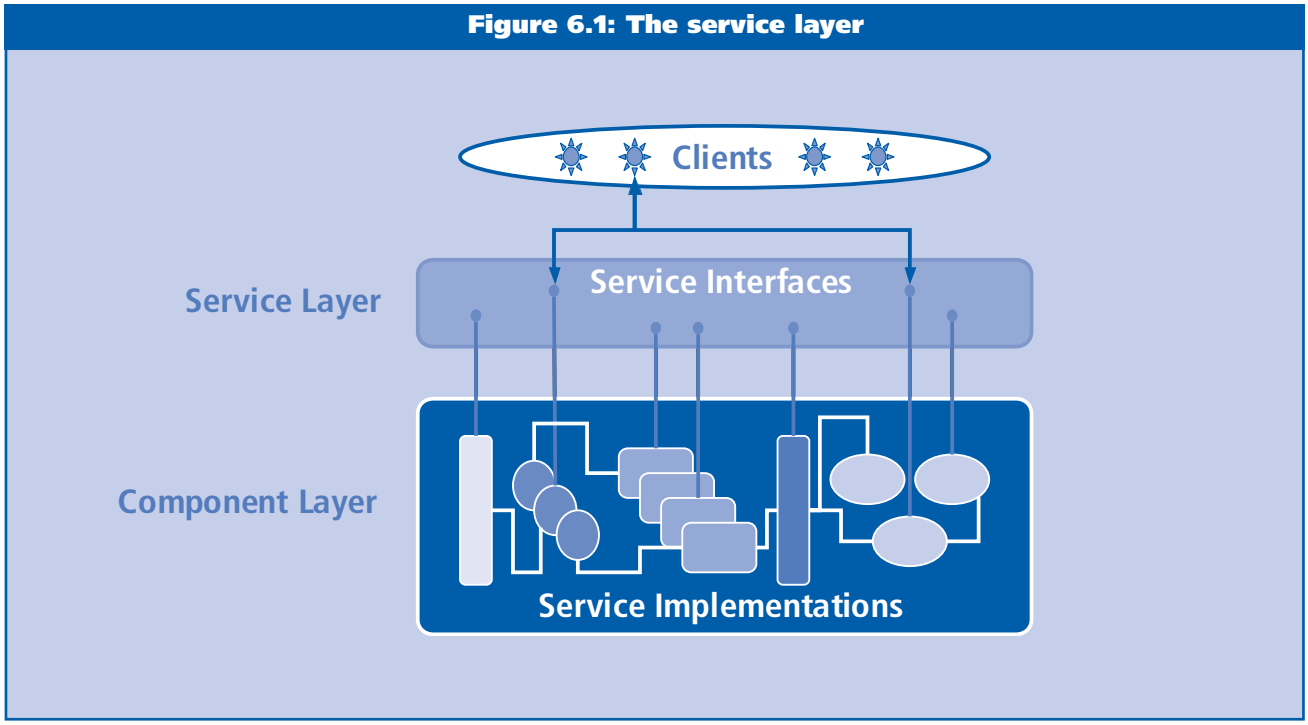
The protocol used for transporting XML standard service messages between clients and service providers is also removed as a dependency by the use of HTTP between Internet nodes. However, this protocol does not satisfy all requirements. The standards approach now adopted is to eliminate this dependency by defining the choice of protocol — WSDL binding information — to be a dynamic or deployment-time issue.

The location of end points in a service network is addressed by WSDL port information and so, like the choice of protocol, this is eliminated as a dependency between clients and service providers by use of dynamic or deployment-time selection. The mechanism for discovery of port and binding choices available in WSDL documents is also defined by standards — specifically UDDI (see *www.oasis-open.org*) and WSIL — to support deployment and run-time use of internal and external service registries by middleware infrastructures.

The remaining important dependency concerns the timing of interactions between clients and service providers. Some may be 'fire-and-forget' — with no implied time dependency. But others might involve critical event notification or 'remote procedure call', interactions — with a concrete timing expectation. Yet more interactions may involve exchange of 'document' messages — with a much more relaxed asynchronous timing dependency.

The standards for eliminating technical dependencies on both reliable messaging and asynchronous styles of interaction (between clients and service providers) have not yet been fully specified. Nevertheless there is a clear intention

Figure 6.1: The service layer



to complete this task (see *www.oasis-open.org*) and to deliver middleware infrastructure products that provide a full range of options.

Service container middleware

When application server products — such as IBM's WebSphere Application Server or BEA's WebLogic Application Server — fully implement the latest J2EE 1.4 specifications, service implementations will be deployed in containers along with other application components with all the advantages of shared resource and management facilities. Multiple services with their interface definitions, ports and bindings may be deployed in each container. In fact, many of those existing application components will be given service WSDL descriptions for access as services but run otherwise unchanged.

In J2EE systems, services may be implemented as JavaBeans, Servlets, Enterprise JavaBeans, process flows and Java Resource Adapters — the latter providing access to:

- **legacy information systems**
- **messaging backbones**
- **off-the-shelf software packages.**

The facilities provided by J2EE containers may also be modified by system providers (Figure 6.3) such as Java Messaging in the usual way. A fully built-out application server configuration will potentially provide rich enterprise service containers.

Service requests will enter containers from a variety of ports using a variety of different bindings — as described by the deployed WSDL documents for services provided — after passing through a 'pipeline' of handlers (Figure 6.3). These handlers will provide a number of standard processing options and provide an opportunity for custom handling of service requests. The container will provide standard service discovery mechanisms for access to WSDL service descriptions on the fly or at startup.

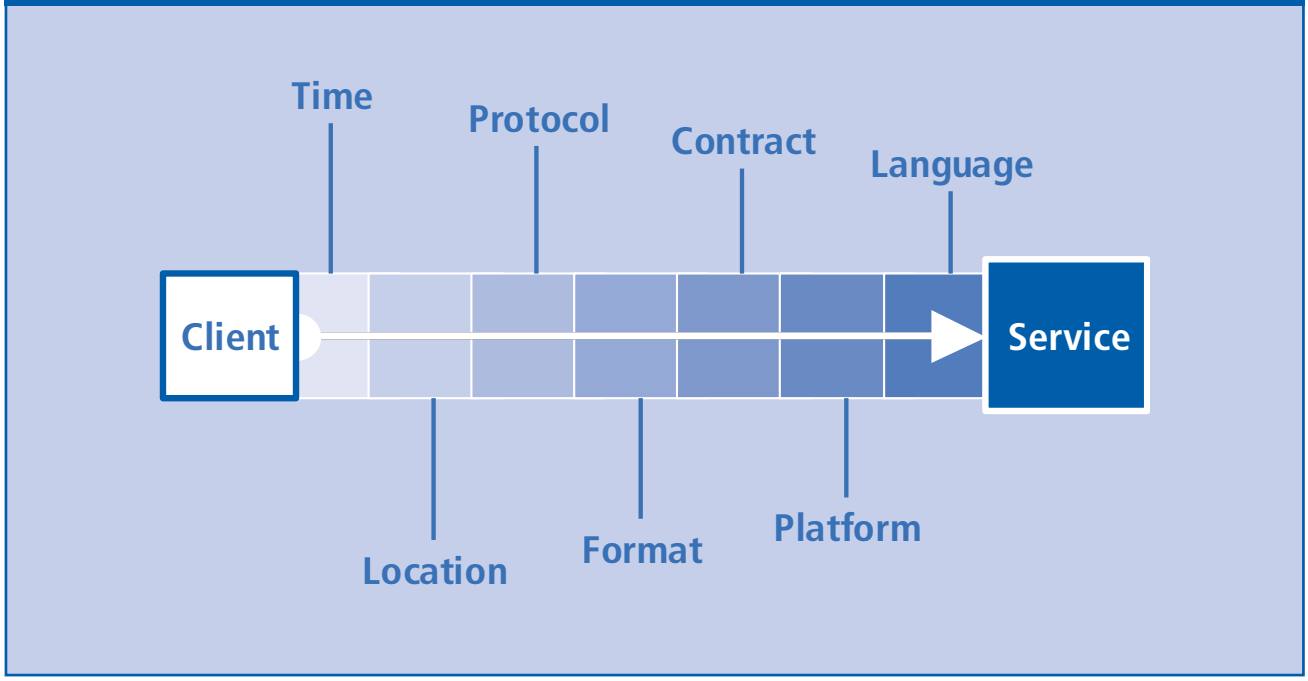
The value to be found in the pipeline can be quite considerable. Both open-source Apache AXIS and J2EE 1.4 JAX-RPC specifications describe this approach for service request handling. Incoming message contents may be:

- **decrypted**
- **digital signatures checked**
- **clients authenticated**
- **policy requirements validated**
- **SOAP header requests honored.**

On the reverse flow, outgoing messages will be:

- **decorated with policy requirements**

Figure 6.2: Dependencies between clients and services



- **logged for audit purposes**
- **digitally signed**
- **encrypted as required.**

Other custom functions — such as data transformation and exception handling — may also be configured into the pipeline. Note that service provider components will often act as clients to external services — and so the pipeline will work in reverse for these components. Even when the client is a standalone component, compliant with J2EE 1.4, there will be a small container and a pipeline of handlers.

This symmetry in system design will prove valuable as it:

- **enables re-use of infrastructure technologies**
- **may be used to enforce enterprise policy for services.**

Furthermore, service containers may provide local optimizations for service requests that can be satisfied within the container, within a cluster of related containers or within a certain enterprise policy domain.

The majority of J2EE vendors will be delivering service containers as extensions to their existing application server products. Much of the infrastructure has already been delivered, for example, by IBM (in the WebSphere Application Server V5 products with its proprietary extensions for wide ranging service access to existing assets through adapters and service gateways). Other vendors are likely to adopt a similar approach.

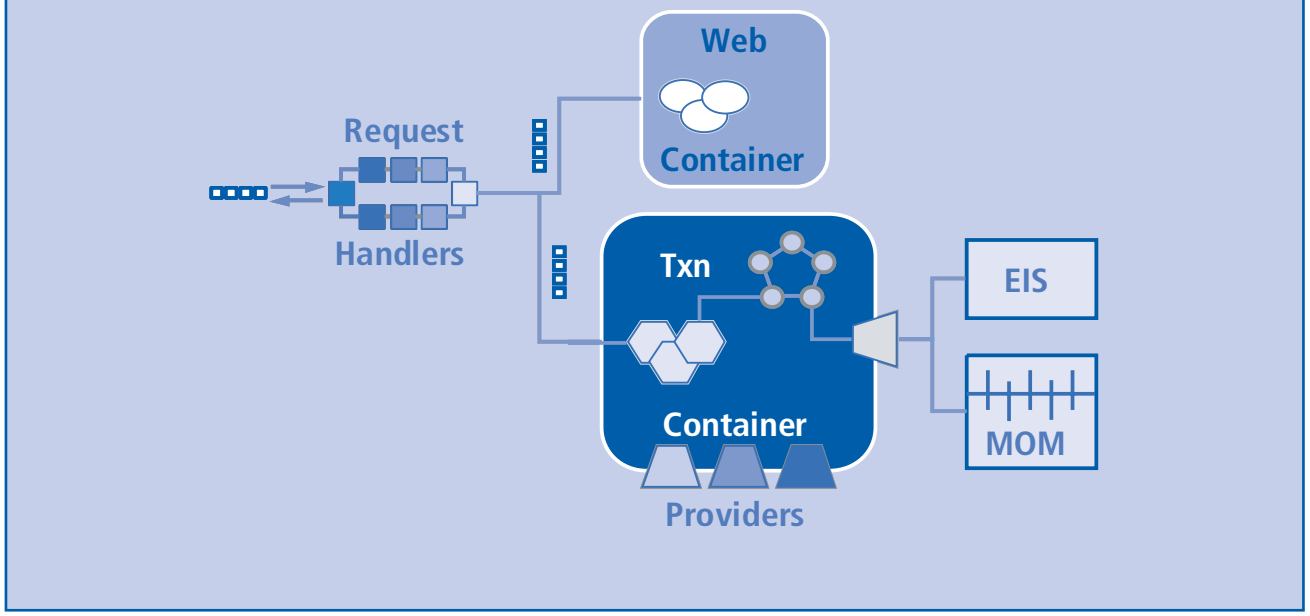
Service bus middleware

The idea behind de-coupling of clients from services has also prompted some thought leaders to propose the use of a bus concept for building middleware infrastructure in support of service-oriented architectures. This concept would facilitate systems architects and implementers to think about any-to-any connectivity for business service components.

The middleware product technology needed for provide such connectivity is logically an extension to technology already provided by many JMS and MOM (message-oriented middleware) vendors. In addition to standard messaging, queuing and protocol support, intelligent routing and broadcast facilities as well as standard access API support, service discovery, service selection and service request handling APIs would be needed.

Various vendors have provided early service bus technology for evaluation in the form of a framework for defining Web Services processors. Using this framework it is possible to define:

Figure 6.3: System providers



- **service clients**
- **providers**
- **gateways**
- **intermediaries**
- **and much more.**

In this context, service clients are supported by bus 'on-ramps' and providers by 'off-ramps' (Figure 6.4). Clients will connect to future service buses using JAX-RPC APIs to send service request messages to a targeted service. The on-ramp may dynamically select alternative ports and bindings for the service using pluggable discovery and selection mechanisms once a message has been received and based on quality of service requirements. The messaging engine then uses chosen binding protocol providers to deliver the messages to corresponding off-ramp channels.

Filters are available in the bus to apply custom or enterprise standard handling of service requests. Such filters may be used to:

- **validate and compress message contents**
- **provide management function, such as metering and billing**
- **deliver enrichment, or suppression, of service request information — as required.**

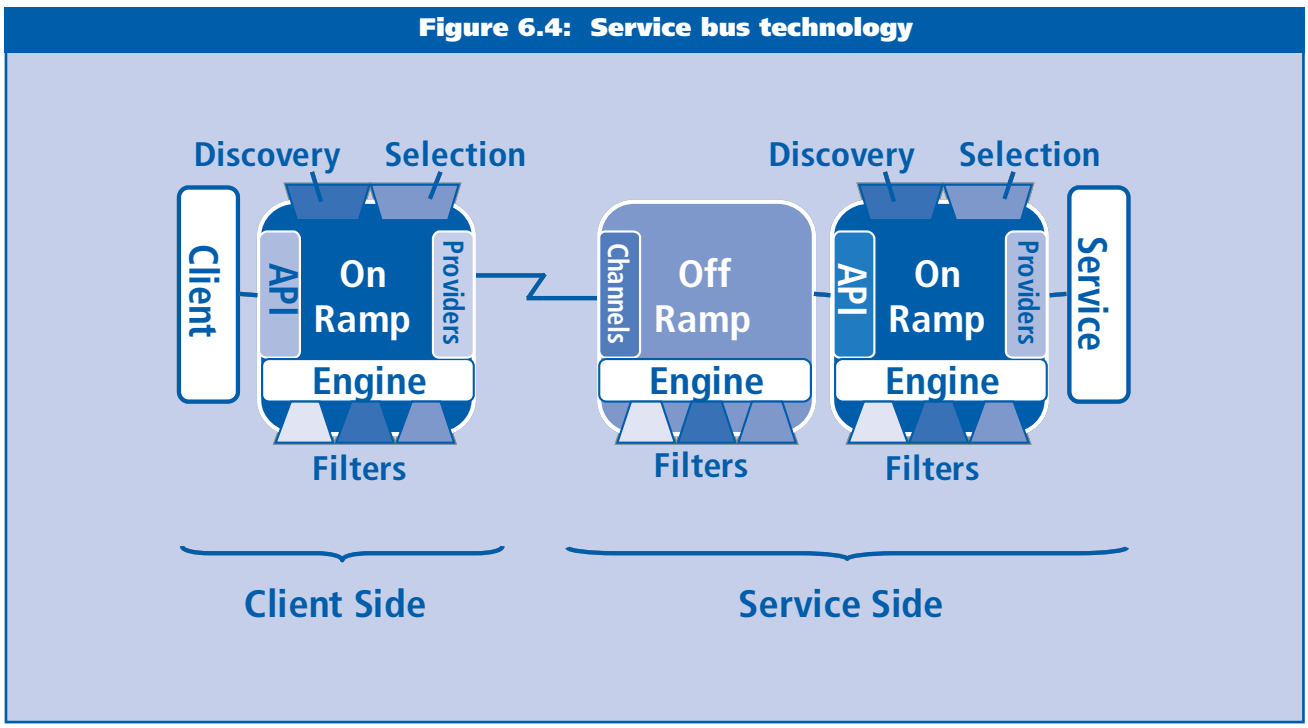
Bus filters are analogous to the pipeline handlers described in service container technology above.

The off-ramp may dynamically publish WSDL descriptions for the services that have been attached to the bus or it may publish service descriptions to UDDI registries for later discovery. Filters may also be applied to service request messages as they are delivered to attached services. Such filters might be used to apply transformations to message contents so that they conform to evolving service descriptions over time.

The value of a service bus comes from the de-coupling it provides between a wide range of service clients and an equally wide range of service providers. Using filters and deployment descriptors, such a bus provides a number of mediation capabilities that isolate clients from changes made to service implementations over time and adapt service information models to provide sophisticated integration between business functions.

Using reliable messaging and standardized security measures, the service bus can provide assured delivery of documents between business processes and the scalability and robustness required by large enterprises. Whilst the service bus will almost certainly be deployed behind corporate firewalls at first, the potential for supporting B2B service interactions is real once reliable messaging and higher level

Figure 6.4: Service bus technology



standards are agreed and implemented.

Building a service infrastructure

As enterprises deploy service-oriented infrastructures, the starting point will often be a requirement to expose existing business functions to new clients — customer, supplier or employee portals — as well as standalone clients and other services acting as clients. J2EE application servers, like WebSphere or WebLogic, may be used to provide:

- **encapsulation of legacy business functions using WSDL service interfaces**
- **the service containers needed at run-time.**

Over time, new services will be developed and off-the-shelf package services will be integrated into the infrastructure and an internal registry of reusable assets will be defined (Figure 6.5). Whilst much of this can be achieved with a single application server, enterprises will often have already deployed multiple platforms that must be integrated. At this point it may be advantageous to introduce a service bus to link the several service containers into a single manageable infrastructure.

Application servers will have the necessary support for service discovery, selection and invocation using a number of different binding protocols and access paths to corporate data resources. However, the deployment of a service bus will introduce additional protocol support, dynamic selection of the best available service endpoints, management to defined policies and service level agreements as well as shared mapping, transformation, logging and other auxiliary services. As standards evolve — such as BPEL (see *www.oasis-open.org*) — service flows will be deployed as points of integration that define higher level business functions. Automation of document transformation, enrichment and logging of service messages will be integrated into the bus but more sophisticated business work flow will require, and enable, service flow state management and human interaction where exceptions are thrown up.

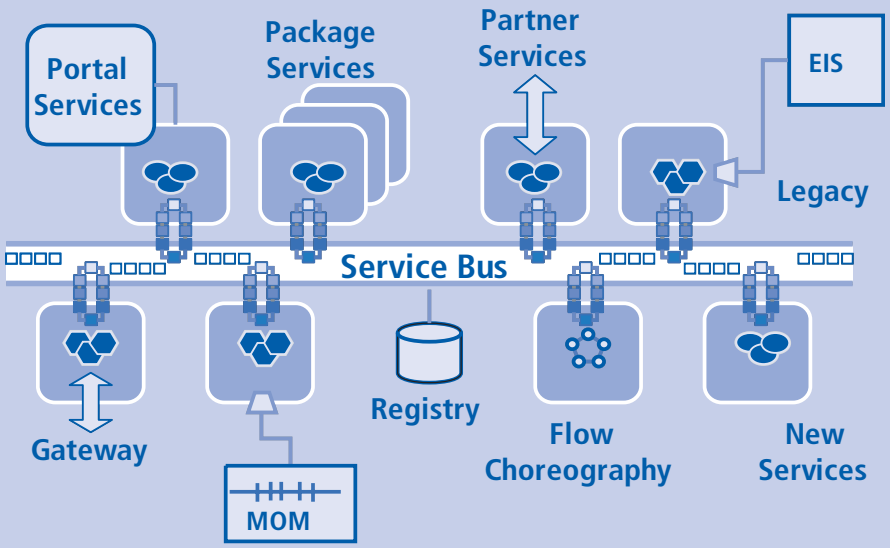
The service bus will also provide gateway connectivity and control for service requests coming from business partners and external users. The same gateway can serve to:

- **control internal access to business partner and Internet services**
- **monitor usage**
- **apply enterprise policies to outbound requests, as required.**

Management conclusion

As enterprises face mounting challenges in the marketplace, IT departments are under pressure to integrate and

Figure 6.5: Converged midleware infrastructure



enhance business processes, reduce costs and improve time-to-market in order to develop or maintain competitive advantage. Well established middleware vendors are leading the industry towards service-oriented architectures for a solution to this increasingly complex scenario.

J2EE application server middleware products are evolving to implement open standard functionality that supports service description, discovery, deployment and invocation in managed containers along with existing application components. Sophisticated tools are also being developed for easily modeling and mapping existing business application function to service interfaces and a wide range of service adapters are becoming available as bridges to valuable legacy systems.

New middleware products are emerging that link service containers in a network to provide highly reliable and scalable transport for standards-based service messages. Products such as Sonic ESB (Progress/Sonic), Tifosi ESB (Fiorano), SpiritWave (Spiritsoft) and Web Messaging Platform (Kenamea) have been made available in recent months as well as the alphaWorks technology from IBM. Unlike existing hub-and-spoke messaging backbones these 'service bus' products are expected to offer simple any-to-any connectivity for services delivered either as extensions to MOM packages or smaller offerings based on a JMS provider with added support for XML messaging. The value add lies in the mediation achieved between clients and service providers where application information and security models require mapping or transformation and in the management of quality of service per defined agreements/policies.

Much of the technology found in J2EE service containers can be re-used and extended to implement the service bus concept. This integration, and re-use of standards-based technologies, can only be good. Although the capabilities of the service bus and container infrastructure are focused on implementing the architectural service layer (Figure 6.1), they could be less expensive to deploy and operate over time as skills and other resources are re-focused.

All of this infrastructure is — at one level — really just plumbing. But it is also a sound investment in the foundation needed for the next architectural layer — the business process layer.

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]