FINANCIAL MIDDLEWARE**SPECTRA**

incorporating Enterprise Middleware

Contents

May 2002

Volume 16 Report 2

Counting the cost of middleware's evolution

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

Management introduction

Middleware is now accepted as a necessity in the financial sector. Few question the need for it to enable:

- ***intra-business application integration (within the financial enterprise)***
- ***inter-business application integration (between two or more different organizations, whether financial or not).***

That said, middleware products still suffer from drawbacks or deficiencies. The most obvious of these remain, unfortunately, its impenetrability, complexity and poor delivery. In part, this is because much of what middleware delivers is abstract and intangible. In part, it is because it comes from the 'systems programming' community.

In this **FINANCIAL MIDDLEWARESPECTRA***, two themes are expressed:*

- ***the gradual raising of function to a level which 'ordinary' can relate to and use***
- ***the need for greater ease of use and clarity.***

Criterion Assurance and Reuters

The case study about Criterion Assurance (page 2) shows work flow being deployed to achieve business expansion in an insurance company. As Chris Booth discusses, such an approach meant that Criterion did not have to expand its premises or numbers of people — of drown in paper as it outgrew its existing premises.

Such benefits, however, only came through determination and the willingness of Criterion's management to be involved. As this case study makes abundantly clear, "process automation is about much more than the steps of the process itself ... it is not just about a third party drawing a flow on a whiteboard, knocking out a few programs and then implementing a shiny new flow."

In a similar vein, Urban Bettag and Mark Curtis of Reuters (page 12) describe a project that Reuters has worked on in recent months to explore how it can make its (Reuters') customers' interactions simpler and less costly. One of the key conclusions from this work — which involved vendors as varied as Oracle, Requisite Technologies, BusgsEye, Bowstreet and Siebel — is that buyers still have to do the integration.

Expecting your middleware products, or even applications which use middleware, to have this already done for you is still vastly over-optimistic. Despite huge investments in function, usability remains a long way behind.

Declarative or procedural programming models

A logical step on from these two user experiences is the argument made by David McGoveran (page 32), namely that the procedural programming model is deficient for business application integration. He explores multiple reasons for this.

In his analysis, the declarative approach is superior — not least because it is simpler to understand. He suggests that if we really want business users to assemble composite processing (made up of processes already captured in applications) we need to accept a declarative approach. If we do not we will be held back by the constraints inherent in the procedural approach.

The Eighth Layer

In a more historical and philosophical piece (page 20), Peter Bernstein reasons that the old communications, and telecommunications, business model has broken down (even though many of its executives are reluctant to admit this). On the other hand, the rise of 'virtual .ME' — our expression of our own, multiple, computing identities — is forcing middleware to the center of focus.

In his vision, "it is middleware which creates the place, processes and states of the Eighth Layer. Hence, understanding the deliverables, and myriad, of new requirements of the next generation of middleware as the electronic proxy for my volition, is the only way for vendors to be successful."

Web Services, Java

Web Services continue to enthuse vendors without having quite the same impact on customers. Mark Lillycrop (page 27) looks at why this is and whether it will change.

On page 40, Keith Jones reviews progress at the Seventh JavaOne jamboree in San Francisco. As he describes, much progress has been made on many fronts — from Java Card to J2ME to J2SE to J2EE, XML and Web Services. Most of this is good, even though the once proclaimed 'write once, run everywhere' has not quite materialized in the way expected or desired.

Free middleware

Finally, Tom Welsh starts the first of a two part examination of 'free middleware'. This one focuses on what you obtain for very little for (or from) the CORBA world.

What he discovers is that you really can have most of what you want for next to nothing — so long as you have the deep technical skills (also known as systems programming or even hacking capabilities) and pockets to cope with what you receive. Arguably this is neither better, nor worse, than with commercial ORB middleware.

Management conclusion

Middleware continues to evolve, and fast. Financial businesses are seeking both business process automation as well as business application integration. Both depend on middleware.

However, as the various analyses in this **FINANCIAL MIDDLEWARESPECTRA** *show, the financial sector must expect to do its own integration. Middleware has not yet risen to the point where business analysts (as opposed to IT professionals, especially systems programmers) can use it. The need is there. The tools are coming. But it remains around the corner.*

Automating business processes at Criterion Assurance

Chris Booth
Managing Consultant
Strategic Thought

Chris Booth has worked on several process automation projects. He has been involved with all the principle activities — from process analysis to process flow to people inclusion to project management. Of his assignments, one stands out as being of more than usual interest. This is Criterion Assurance, a German owned company (part of the largest German mutual assurer, Vereinighte Haftpflichtversicherung VaG — or VHP), that is based in Petersfield, England and which has links to other European insurers.

Strategic Thought is a consulting, system integration and software company based in Wimbledon, London. It was founded in the mid-1980s to focus on database implementation, particularly involving Ingres. Work with Tuxedo and ENCINA OLTP middleware was followed by involvement with IBM's WMQ (and its associated family) as the market for OLTP middleware showed few signs of developing further. In addition, Strategic Thought produces a risk management product — called Active Risk Manager. This was used by Lockheed Martin (in association with its successful award of the Joint Strike Fighter project) and has been deployed at Rolls Royce, BT, AMEC and elsewhere.

In this case study, Mr. Booth discusses:

- ***why Criterion chose to introduce process management to expand its business***
- ***the activities that were involved***
- ***the lessons learned.***

Criterion Assurance and its business model

Let me start by describing Criterion Assurance. It is an insurance company which sells life assurance and pension management — but it does not sell direct to the public. It sells through a network of brokers.

The added value which Criterion brings is that it offers insurance processing to brokers so that these brokers can sell insurance products under their own name to their customers. Criterion processes the applications, does the underwriting, deals with claims, handles the administration and produces all the documentation — under the broker's name and on its letterhead.

An extension of this is its 'club arrangements': if a sports or some other form of club or association wants to offer life insurance to its members (as a side benefit of being a member), Criterion will handle everything for the club.

In terms of size, Criterion is relatively small, with only about one hundred staff. One reason is that it does not have the huge sales force that most insurers seem to think they need.

In terms of history, which leads to why process automation was needed, the Company has been around for some 30-40 years. Until quite recently it had been largely dormant — running off outstanding policies. Then it was bought by the German insurance company VHP, which hired an executive called Nigel Cooke to head it up and to investigate what might be done.

He saw the potential in the Company to use its substantial experience in administering insurance and pension policies. He recognized the value, as well as the opportunity, for others (for example, brokers) to exploit Criterion's insurance licence.

His new business model envisioned Criterion making money on the underwriting when the broker produces an insurance proposition: it would quote a price to the broker; if accepted by the broker's customer, Criterion would underwrite the policy and perform the associated administration on behalf of the broker. Furthermore, the arrangements with each broker could be different, making each arrangement look personal as well as fitting the size of broker.

The attraction to brokers, besides the personalization, was that they would be able to expand their businesses without having to undertake people intensive administration. They would contract Criterion to do this.

VHP invested in this business model and, over the past three to four years, Criterion has been extremely successful. Furthermore, the model has been broadened to reach beyond the UK. For example, Criterion has arrangements with other overseas European Union insurers which use Criterion for their back office processing and so are able to offer more attractive products to their customers in their own countries.

Growth and success

From an operational point of view this model is much more complex than the apparently simple arrangement with a broker implies. For example, for each quotation of an underwriting price, it is essential that Criterion is sure that it has all the application details and knows where policy information needs to be sent as well as how subsequent contacts are supposed to occur.

In addition, Criterion operates its own call center. Sometimes calls come directly into this from a policy holder; in other circumstances, it may be the broker who makes contact. This will vary from broker to broker depending on the size of the broker and how much it wants to outsource to Criterion.

Such has been the success of this approach that Criterion found itself taking on more and more staff. Before long it had to face the prospect that it was going to reach the point where it was likely to run out of space in its existing building. In addition, and as worrying, was that the volume of paper was constantly mounting. If growth continued, it was clearly going to reach a point where something would need to happen if expansion was to continue.

One choice was to move premises, hire more staff and deal with (and drown in) ever more paper. The preferred option was to find a way to automate the Company's processes so that the existing premises and staff would be sufficient to handle the increased business.

Investing in some form of process flow automation was a logical extension of the 'green screen' software that was already in use. This ran, and is still running, an application called Slogans (from New Zealand). It is hosted on HP-UX and offers:

- **underwriting**
- **claims processing**
- **account management**
- **the management of customer names and addresses**
- **etc.**

The down side, to Slogans, was that it is written in a language that is proprietary to the software company that created it. As a result, Criterion employs specialized developers who are responsible for the maintenance and evolution of the functions at Criterion. These are scarce and skilled people, a factor which broadly determined that some form of application integration and process automation would be needed. There was no intention to replace what already worked.

At the same time another application indicated that application integration would need to be a necessary part of any process automation. Criterion had bought an image system for scanning and holding document images. This was loosely linked to Slogans, using proprietary software (also sourced from New Zealand) written in VisualBasic. While this was relatively easy to maintain, it was not particularly easy to use. It was also in danger of not being used and this partly explained why the paper was mounting up.

Matters reached a point, therefore, where improving and automating business processes became the obvious way to support continued growth that at times was hitting 300% a year. The danger was of reaching a point where the new business existed (with brokers wanting to buy) but Criterion might not have the means to accept it.

Action

The first step was to bring in some process expertise. Once the decision had been made, that process improvement made sense, two possible ways forward were investigated. The first was with Staffware and the second with a combination of IBM and Strategic Thought. (IBM wanted us involved because we had experience of its products and were of a similar size to Criterion.)

Initially Staffware won the deal, apparently on a claim that process automation across all the Company's primary activities could be completed in three months. We heard about this and went back to see Criterion, with IBM. Jointly we explained that process automation is not done so simply or so fast. Instead we offered the view that delivering the first enterprise process flow, including the necessary application integration, would take a minimum of three to four months. The others would then follow over a period of a further year or so, depending on how fast Criterion wished this to happen.

Criterion asked why we thought this. We explained that process automation is about much more than the steps of the process itself (as I will discuss later). It is not just about 'a third party drawing a flow on a whiteboard, knocking out a few programs and then implementing a shiny new flow'. Sufficient time to consider what needs to be delivered is required, if success is to occur.

Criterion considered this and our assertion that it would need to make its staff available in order to obtain success. It accepted that it would need to engage and its staff would be a key component if it was to to ensure that the resultant flows really did match what its business needed for the future. The pendulum swung — and both IBM and we were invited to work at Criterion.

The foundations of the solution

In terms of the primary hardware and software, Criterion purchased:

- **a UNIX server (AIX)**
- **MQSeries Workflow**
- **Content Manager (for scanning and managing the document images)**
- **WebSphere Application Server (to run the Java bean functions)**
- **WebSphere MQSeries Integrator (now shortened to WMQI)**
- **Websphere MQ (or WMQ, what used to be called MQSeries) for messaging between Slogans and the other applications (including WMQI and MQSeries Workflow).**

One attraction of all this was that IBM was the single source. If there were system or software problems, it was clear where to go. We (Strategic Thought) were commissioned to deliver the initial integration and process flow automation, using IBM's products and tools.

Implementing process automation

To start with, the impetus lay with us. Criterion's IT department is small and was fully occupied in keeping Slogans running. In terms of immediate skills it did not have those needed for the WebSphere/MQSeries products. We agreed that we should supply the necessary system expertise — including the initial installation and configuration of the new hardware and software — until Criterion was ready to take over the operational responsibilities.

On the process automation front, as we had explained, there was no way that Criterion could avoid its staff being intimately involved. We, as outsiders, could not tell it (Criterion) what was best for its business. Its staff understood far better than we ever would what each of its business processes accomplished and how. On the other hand what

we could do was facilitate progress in order to minimize the amount of time that Criterion staff needed to commit.

The point here is that any organization undertaking process automation has to make that commitment of staff. This commitment is substantial, and it was impressive that a company of Criterion's relatively modest size was prepared (and able) to do this. Nevertheless it had understood, and we jointly put together what we called the 'business process team'. There were three of their people involved as we focused on the process analysis. In addition, it hired one person with Java skills and brought one person over from the Slogans team to assist whenever we moved into areas relating to its existing IT.

At this juncture another key decision was made. This was to go for one initial process flow (with the others to follow later, when the first was working and accepted). As part of the pre-bid work, Criterion had examined its business as a whole and identified:

- **ten primary processes — covering activities like processing claims, taking on new business, sending out letters and so on**
- **those ones which were causing the greatest pain.**

Of these, taking on new business was the one which needed the most urgent automation. It was this that would constrain growth if it was not accommodated. In addition, addressing this first would create the time to look at the other areas later. So that was where we began.

Defining the first process flow

Over the first month the business process team took the initial Criterion analysis and began to question and validate that analysis. Using a workshop format we had the Criterion people dedicated from 10am-3pm (deliberately leaving them time for their regular activities outside these hours).

In the workshop we started off by modeling what its people thought the 'new business process' was. In itself this was interesting, particularly with people from different areas of the operation who did not necessarily know what others did (or why). Just discussing what actually occurred was an eye opener, and not just for us — which makes a telling point: process automation will not work, either in isolation or abstraction.

Initially we did not attempt the modeling in MQSeries Workflow itself. The risk was that more effort would be put into learning to interpret what the tools produced than in identifying what really needed to occur in the 'new business process'. Instead we used traditional white boards and pieces of paper stuck on the wall. This was also easier to amend as well as to understand.

Once this was complete we moved on to look at what could be improved and what was inefficient about the existing process. For example, Criterion had a classic problem with the direct debits from customers' bank accounts. With a direct debit you have to give the account owner 14 days notice before you change a payment value. But the existing 'new business process' took so long to complete that new customers had to be told that two payments would be taken in 28 days. This was confusing, costly in customer relations terms and poor for cash flow.

What we set out to do was find each sub-process problem area and look for a way to eliminate (or reduce) it. In effect we were looking for the standard rules which could be applied to automate and guarantee the 'new business process' of underwriting. We did not, however, attempt to solve everything immediately. Instead we opted to define one quick, simple process that:

- **covered 80% of the 'new business process' activity**
- **handled exceptions (the other 20%) manually, at least to start with.**

Rather than trying to deliver the complete process, we looked at what rules would apply to ensure that 80% of the work would complete without manual interaction. We then built this, with the intention that once it was working we could go back and gradually whittle down the remaining 20% — by adding new rules and actions — until only a few exceptions were left (those where it did not make economic sense to pay for a software solution).

As part of this we undertook much work with the prospective users to ensure that whatever we came up with actually reflected how they wanted to work (as well as how Criterion wanted them to work). This is important. Frequently we discovered that the traditional form on the desk and their use of the green screen (the access to Slogans) bore little relation to the way that the day to day process actually worked.

Much time was also invested in looking for ways that reduced people's effort as well as automating the flow. If it was useful, we reasoned (using experience from other projects) that people would be more likely to accept the changes in working procedures and operations.

What helped us here was that we approached the new business process as being one flow or process, rather than a master flow with hordes of micro or sub flows. There were two or three significant sub-flows but, because the Slogans system was very good at doing the data processing, we exploited its efficiency to reduce the overall process automation aspects from first flows.

Integrating applications: mapping the business to the technology

To work with the Slogans system required us to write a specific application adapter. This was needed so that we could send, and receive, information for the underwriting process — age, date of birth, smoking habits, health record and all the other data that was needed for Slogans to decide whether to offer a price for the risk to be insured. What we were doing was:

- **placing a request for risk assessment/pricing on a queue (from MQSeries Workflow)**
- **transporting that to Slogans (via WMQ)**
- **having Slogans make the appropriate calculations**
- **placing the response on an outbound queue to go back to the user (WMQ)**
- **MQSeries Workflow picking up the response off that queue and presenting this back into the 'new business process' flow — and continuing the flow.**

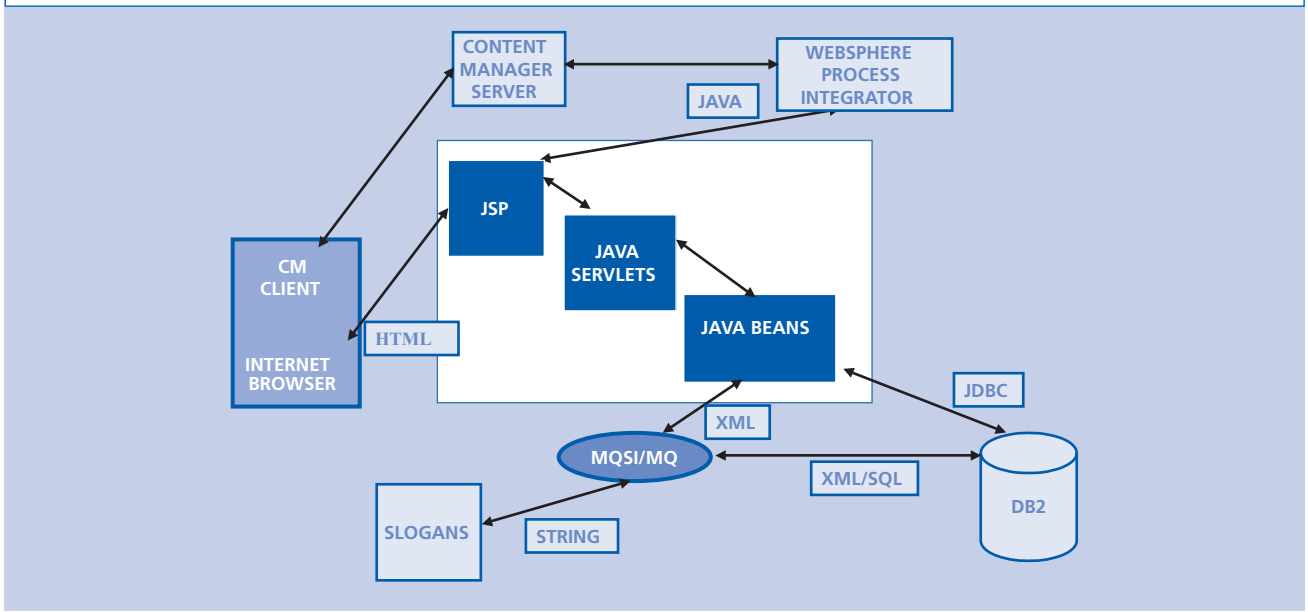
In this way we could 'hide' Slogans from users, without depriving them of its functions and while still making the process easier to use. The downside came with this adapter. Adapters can be complex to develop; in principle, pre-built adapters are preferable if they are available.

In the Criterion instance the issue was about Slogans. It is not an application where adapters already exist (as they do for the likes of Oracle Financials or PeopleSoft or JD Edwards). This meant that our integration effort had to understand what, and how, Slogans worked. In practice, we worked with the Slogans' developers (in New Zealand) to create a specific Application Programming Interface (API) that we could use.

This also flags another issue. Up until we began the work on the adapter we were working at a high, business, level. This focused on identifying the overall flow. There were, however, two 'lower' (as in IT-related) levels. These identified the data needed to satisfy a flow plus where this data existed, where it had to go and how.

Only when you have resolved these latter levels of detail are you able to start construction of the automation of flows. Looking back I am not really sure that Criterion had wholly appreciated that there was as much to automating process flows as was involved. Its people certainly knew about the top level (the business process). It was the need to think about the two lower (IT) layers that is not obvious (until you have been initiated).

Figure 1.1: Custom-build EJBs



That said, it made no difference. Criterion knew what it wanted and was quite prepared to support the initiative.

The key point, we think, about process automation type of projects is that they are much bigger than just the apparent business process. They require you to dig deeper. You have to address both the big business issues and the (seemingly) less significant ones — those relating to technical issues (like not losing data in transit and where information is located). You have also to worry in detail about negative or exception situations: what happens if (say) underwriting returns a negative answer. What does Criterion (and the broker) do now?

Consideration of such a diversity of issues cannot be avoided. On the other hand it really clarifies what the business is doing, and why.

Development

Once we had the 'new business process', and the underlying two technology levels, defined we could start the development. Relatively speaking — the adapter excepted — this was pretty straightforward.

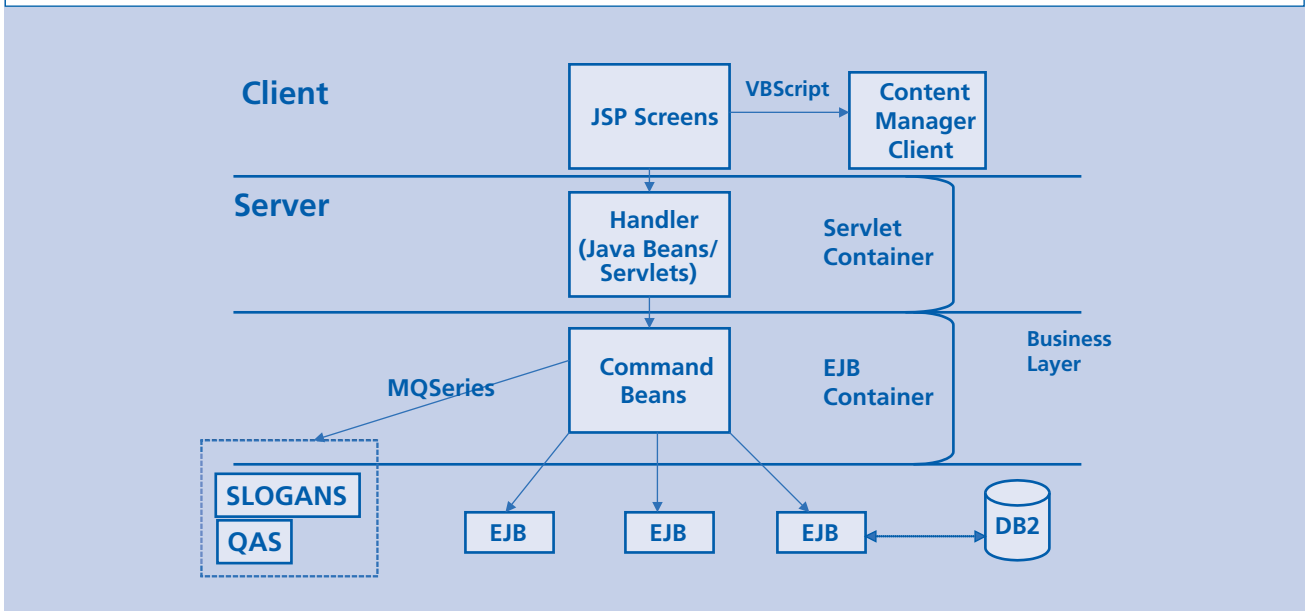
Yet, even here we had two parts. This was because of the imaging and the need to integrate Content Manager with both Slogans and MQSeries Workflow. The basic issue was ensuring that imaged data was appropriately indexed in such a way that both Slogans and, most importantly, MQSeries Workflow could exploit the electronic images rather than having to use paper.

To cut the story short we held another workshop to define the indexing structure, in effect to describe all the ways in which Criterion might wish to access the images (by customer, policy number, broker, etc.). This enabled us to create the appropriate structure within Content Manager which could then be integrated with Slogans and MQSeries Workflow.

For the development of the flow itself, we had decided upon a Web-type interface — a browser. Users would access MQSeries Workflow through a browser session that was hosted by WebSphere Application Server. The latter ran some custom-built Enterprise Java Beans (Figures 1.17 and 1.18) on the AIX machine that provided the access method for working with the MQSeries Workflow flow. As each stage of a flow progressed, the browser was updated.

By doing this we delivered one browser image on the desktop. This not only simplified installation but it gave a consistency that had not been possible before. In addition, although not one of the immediate objectives, it opened a way for third parties — Criterion's brokers— to enter details direct into the new business flow. Providing this not only benefits the broker but brings Criterion even more process efficiencies. Its staff do not need to be directly involved (unless needed) while the broker obtains an even faster turnaround.

Figure 1.2: Architecture



If the business contribution is easy to describe, developing what users wanted was not quite so simple. We spent much time reviewing the presentation logic with users in order to understand how they wanted to work. We actually ended up with screens looking much like the application forms. This was entirely logical as the form had been designed from the same set of premises.

We then wrote the link to Content Manager so that the browser could call up images. Indeed, we went further. When the mail arrives each morning, each document is now:

- **scanned, by the mail room**
- **loads some basic index information (for example, the name of the customer and the type of document — whether it is a new application, a renewal or whatever)**
- **initiates a new business work flow**
- **appears on the relevant person's action list (via the browser) for today.**

In this way we removed the paper from the desk. It no longer reached the desk because it was scanned, stored and then used to start the 'new business process' flows which appeared in the browser sessions.

The overall 'new business process' flow, from when we started, took about four months to deliver. That confirmed our original thoughts, especially as we had much education and familiarization to achieve along the way. The Staffware 'estimates' (if accurate) were hopelessly optimistic — unless it had assumed that it could stop Criterion functioning while the analyses were done (in which case there might have been no business left to automate).

The second flow

The second flow was different. By this time Criterion was familiar with what was needed. What it wanted to do was commence a transition so that it could define and build its own subsequent flows. This had always been its intention.

To achieve this we changed the focus of the second flow project. We developed a simple flow which essentially had only one process. We called it the 'One Step Process'. It basically went from step A to B to C to D in a straight line (with no sub-processes).

If this seems odd, our joint thinking was that it would provide a 'pattern' for plugging in other functions and pieces of work for each subsequent flow. Rather than spending four months on each subsequent flow we were enabling Criterion to add process elements when and as it needed them (and when it had the resources available). Another aim of this One Step Process was that we could roll it out to everybody in the business and they could use it immediately.

What did it do? In practical terms, especially compared to the new business process, very little. That was the point. It just brought up a screen saying that you should do a particular sort of task. When you completed this, you clicked on it and then it told you what to do next. And so on. It did not have integrated links to back end systems. The aim was to accustom the rest of the business to using MQSeries Workflow and its browser access.

The results were as anticipated. The business users became familiar with using MQSeries Workflow concepts. By accomplishing this we significantly softened the cultural impact on Criterion of implementing subsequent work flows. People became used to the concept of using process flows, communicated via the browser, to assist them to do their jobs. It also brought forward the time when we could hand over to Criterion for future enhancements.

The third flow

The value of the One Step flow was proved when Criterion reached the third flow. With its people familiar with the concepts, it proved much more straightforward to modify and extend the One Step Process. Now we could step back. Criterion staff could develop the next flows. The state at present is that it is adopting and enhancing the One Step Process to do what matters most for its business now.

WMQI

The work on the 'new business process' did not stop once it was implemented inside Criterion. As I commented earlier, one attraction was to enable Criterion's customers (the brokers in this instance) to have direct access. If this could be automated, then Criterion would improve the speed of processing to its new business customers as well as increase its capability to service additional business.

This is where WMQI has the potential to fit (this is still being researched). It could be the hub into which customers come (off the Web). WMQI would then transform and route messages to the appropriate processing — as well as manage the return messages. This has particular attractions for expanding the international side of the business. For example, one of its overseas customers currently sends Criterion a flat file. WMQI can analyze that file, submit the individual records and compose a return file as a response.

Lessons learned

Despite what many say, process flow automation (or work flow) is not simple if you want to introduce it properly for the benefit of the business. Picking up the existing processes and automating these was not what Criterion wanted. It wanted a mechanism to continue its growth which used automation to improve how and what it did. That is what it achieved, by thinking through all dimensions of the business and technical challenge.

To deliver work flow, you have to engage properly, which almost inevitably means that the initial costs will not be trivial. Whoever does it — whether you pick two or three people or flows — the enterprise has to understand that time and resources are required. Impatience is a mortal enemy. Furthermore, automating processes is not just a technical issue: it is a 'whole business issue'. If you only look at one dimension you will deliver a lopsided solution.

Staff at all levels in all parts of a process need to invest time and effort in order to understand what exists and suggest how this can be achieved (they almost always know best). What you cannot do is 'just' leave it up to either the business process team or to the IT department. Separately, both will get it wrong. Together they should get it right.

To deliver process automation, we recommend that there is somebody senior enough who will sponsor the action and who can authorize the disruption — for example assigning key staff members to the project for most of a day for six weeks or whatever is required. Furthermore, we think that this executive has to be ready to take decisions.

Inevitably, in your planned flows, you will arrive at points where two or more different paths are equally valid, although with different downstream consequences in business (or technical) terms. Someone needs to make a decision, if only so that progress can continue.

We were fortunate at Criterion that its CEO was, on a pre-planned basis, available to make those decisions. That is one reason why we did not fall behind.

In technical terms, the principle lesson we learned was about application integration. Existing back ends play a fundamental role (unless you are recreating applications from scratch — which is too risky for most). Whatever systems you need to work with require thorough thought (and as early as possible); you must think about how you are going to access them.

As I said earlier, for the access to the Slogans system we had to have an API especially written for us in New Zealand. Only then were we able to write the required adapter.

Management conclusion

Criterion Assurance faced a familiar problem associated with business success. It was growing so fast that it needed to do something urgent and different if that growth was to be sustained. Rather than expand its workforce (and paper volumes) it chose process automation, arguably the most sophisticated of the Five Axes, although this depended vitally on all four other Axes.

The two most important conclusions to be drawn from Criterion's example are that process automation must involve people, from those who operate the processes being automated through to management being prepared to make decisions. Flow automation must reach out across the business. Without this, neither work flow nor process flow automation will deliver. Indeed, the reverse is more probable — that such initiatives will become lopsided at a substantial cost. This is unnecessary but it is all too easy for it to happen if the commitment and involvement are lacking at any level.

Business Process Automation (BPA) at Reuters

Urban Bettag and Mark Curtis
e-Commerce Architects, Reuters

Management introduction

Reuters core business is the provision of information and transactional services — from viewing to doing. As a global information, news and technology group, Reuters supplies business news and value-added services to more than 660,000 users in more than 20 languages via over 1400 websites worldwide. In addition, Reuters plays a significant role in the functioning of the financial markets.

Reuters mission is both to provide excellent service to its customers and to make financial markets work on the Internet on a global scale. Urban Bettag and Mark Curtis are in the Reuters Research & Standards Group — the advanced technology stream of Reuters Chief Technology Office. One of the objectives of this group is to bring innovation into the Reuters business through technology leadership and research into strategy, business and emerging technologies. Mr. Bettag's research topics include Web Services, their architectures and how they interoperate while Mr. Curtis focuses on trusted business to business relationships and the supporting technology infrastructure, for example PKI.

In this analysis, Mr. Bettag and Mr. Curtis describe the notion of Business Process Automation (BPA) — the dynamic provision of personalized Web Services — as they envision it applying to Reuters largest clients. They explore:

- ***how this reduces the costs of doing business***
- ***how this delivers improved services for both Reuters and its customers***
- ***the software used, including Oracle's i-Procurement, Requisite Technologies' e-Merge and BugsEye, Bowstreet's Business Web Factory and Siebel's CRM***
- ***recommendations, and lessons learned.***

Business context

One trend being seen across a diverse range of businesses is the desire to move from a product to a customer centric organization. In essence, this encapsulates the desire to put customers at the center of any business. The implications, if the results are to be successful, are both significant and far reaching for all parts and levels of an organization.

From a B2B business administration perspective, a common customer complaint is that the diversity of business processes and practices is excessively complex. The consequent customer message is that 'making it easy to do business' is a step in the right direction.

One way to deliver this is via a set of streamlined business processes that can map to each customer's processes. This would provide a transparent and barrier free business interface.

At the same time, the current global slowdown (if not recession) is fueling a desire to reduce internal costs. Procurement of direct and indirect materials is a major cost, especially among service-oriented organizations. According to the Harvard Business Review, the average cost of procurement can be as much as 35% of the total spend (from ***A Smarter Way To Buy***, June 2001). Furthermore, bringing down procurement costs can have a dramatic impact on the bottom line: a 5% cut can translate into a 30% jump in profits.

In this context, Reuters products and services are frequently one of the top three expenses in major accounts. Whether it is Reuters or the customer which acts first, the scale of such expenditures means that these costs are a natural focus point when customers consider how to cut or control expenses and streamline business processes. Realizing the implications of this has enabled Reuters to look at what it can do to assist its customers, and how such improvements might be delivered.

Electronic procurement

In terms of electronic procurement, the first wave of Internet-enabled procurement can now be considered as largely complete. Most major corporations have finally mastered their first internal procurement projects and are just about seeing the first pay-offs of those investments.

With success, corporations are beginning to look at what further business opportunities exist to cut costs by linking their procurement systems into the back end systems of their strategic business partners. This represents a major change because it involves inter-organization rather than intra-organization considerations. These are necessarily more complex.

As has been proved in intra-organization initiatives, linking business systems together can end up as a Pyrrhic victory if done wrong. Both sell-side and buy-side organizations have to cope with diverse technical and functional variety.

For instance, organizations which have their own e-Procurement or IT Asset Management systems, and require programmatic access to buy- and sell-side business processes via an open commerce service, increasingly wish to obtain these via increased business process automation. A pivotal challenge in such a scenario is providing the scalable infrastructure that is capable of supporting multiple trading partners in an integrated manner (Figure 2.1).

Business process automation

Within Reuters and its customers there is a strong appetite for such business process automation. In some ways it can be seen as 'killing two birds with one stone' — because it:

- **simplifies (and enriches) the customer experience of doing business**
- **offers the potential to reduce (Reuters own) processing costs.**

From a technology perspective we are seeing technologies — like Web Services — and standards — such as ebXML and UDDI — emerging. To us these possess the potential to provide a framework for the delivery of business process automation solutions. The combination of customer desires — when combined with the mutual acceptance of emerging standards-based solutions — suggests that organizations are (or should be) starting to plan for the introduction of business process automation based on these capabilities.

In this context, the Reuters Business Process Automation (BPA) experiment has developed a future vision of BPA. It has accomplished this by working with customers to identify the value and shape of requirements, as well as the technology and standards issues involved.

Understanding the Reuters perspective

The Reuters customer base is broadly organized into a set of customer segments. A customer segment is a subdivision of the Reuters Financial Services market place and divides into discrete customer groups sharing similar characteristics.

The purpose of this segmentation is driven by different customer interests and needs. Within and across these segments there exist a range of relationship dynamics and economic factors which influence the shape of the delivery channel required to serve each type of customer.

Thus, one size does not fit all. In fact each segment may require multiple channels to service fully each customer's requirements.

At present Reuters supports five channels (Figure 2.2) into its customer segments:

- **Individual Direct (B2C)**
- **Business Direct (B2B)**
- **Channel Partnerships (B2B2B/B2B2C)**
- **Consultative Accounts (B2B)**
- **Focus Accounts (B2B).**

Reuters maintains a different variety of relationships with each customer segment. For instance, there is a single relationship with the potential end user within the Individual Direct segment. The amount of variability for the Business Direct segment is limited, which generally means 3-5 forms of relationship cover a large number of businesses. The same applies to the Channel Partnership model, as the number of channel partners (like Charles Schwab) is rather small.

Focus and Consultative Accounts maintain, for almost every individual Reuters customer, a dedicated relationship. Yet, necessarily, every relationship is different. Hence, the degree of variability for both customer segments remains high.

One of the objectives of this customer segmentation is to make it easy for customers to do business with Reuters. This is particularly crucial for the Focus Accounts that purchase hardware, software and digital services in large quantities. Thus one of the general guiding principles applied to these Focus Accounts is the improvement, via automation, of the relevant business processes.

These are the processes operating between Reuters and the customer. Delivery is accomplished:

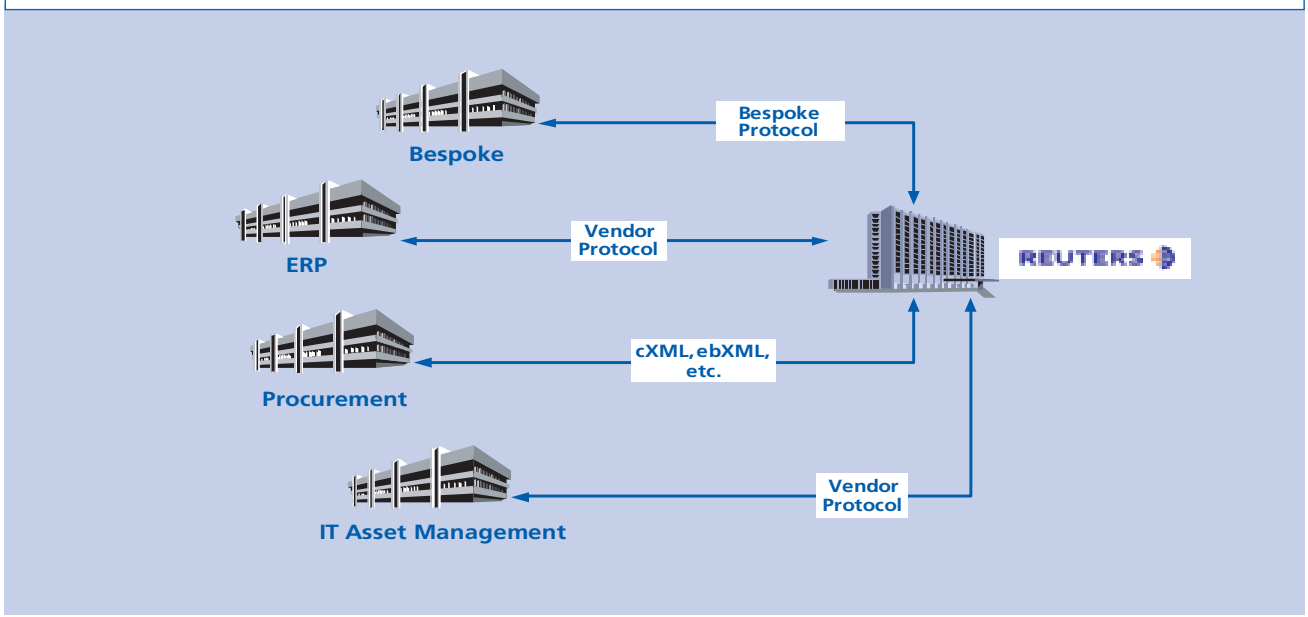
- **either by using Application Programming Interfaces (APIs)**
- **or by introducing straight through processing (STP).**

For the larger, more sophisticated customers the interest is in integrating their own procurement and asset management systems and our (Reuters) own systems, again using APIs and/or straight through processing.

In practice

In practice, within a Focus Account, there is an e-Procurement package — such as:

Figure 2.1: Linking business systems together



- **Oracle's i-Procurement**
- **SAP**
- **Ariba.**

These are accessed by an authorized user via the company Intranet. An internal approval workflow, which varies among Focus Accounts, is integrated to back end systems (like General Ledger or Asset Management). Authorized users can then navigate and search through the Reuters product catalog.

Once the appropriate product has been identified the user will add the item to his purchase order. After the purchase order has been authorized the order is sent to Reuters. Once received by Reuters the purchase order will trigger a set of further actions, such as credit rating, invoicing and provisioning of the service.

In addition to standard purchase instructions, most large accounts have a significant amount of 'churn' in their environments. Inevitably, individual users leave or move within their organization. This results in a large amount of cancellation, addition or transfer of service instructions generated by the customer and communicated to Reuters.

In our view, keeping the product or service catalog up to date while still making it easy for customers to buy is frequently underestimated.

The Reuters BPA architecture

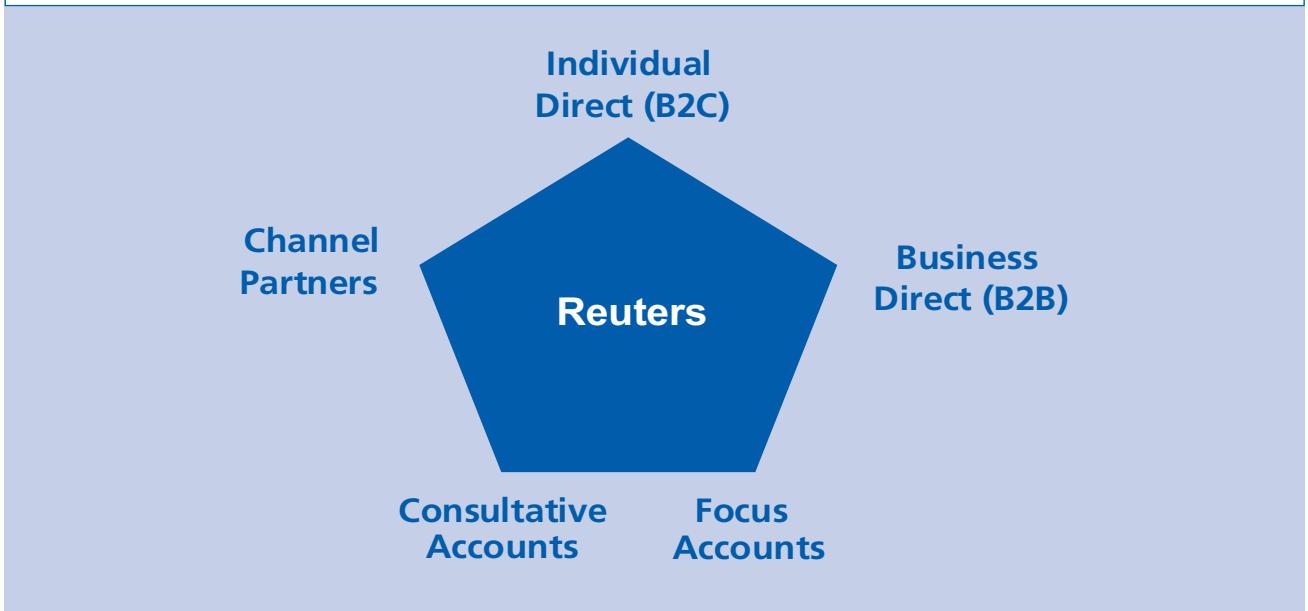
At a macro-level the BPA architecture can be partitioned into three major tiers (Figure 2.3):

- **the client tier represents the clients' procurement and IT asset management solution**
- **the second tier consists of a range of business Web Service; business Web Services — in contrast to micro Web Services — are coarse grain and mainly designed for B2B integration; for service provisioning they consume fine grain micro Web Services, usually Web Service enabled COTS-based systems**
- **the third tier is the Enterprise Information System (EIS) tier; this tier includes a variety of COTS software packages, such as customer relationship management, service permission, billing and payments, authentication, etc.**

The following list of business Web Services represents the core of the Reuters BPA commerce service assembly:

- **e-Procurement: this encompasses the requisitioning, approving, purchasing, checking and receiving of products and services over the Internet; authorized employees requisition relevant products, either for themselves or for another user**

Figure 2.2: Channels into the customer base



- **catalog: this contains the structure and content of all the Reuters products and services; it includes the capabilities for authoring, packaging and indexing of products**
- **deployment: there are many catalog hosting scenarios — whether hosted locally in a customer eProcurement solution or by a third party exchange (or marketplace) or by our own one — with a customer 'punching out' of their eProcurement systems to access the Reuters service**
- **collaboration profiles: the collaboration profiling service represents all the Focus Accounts capabilities for exchanging electronic messages; for instance, a collaboration profile for a Focus Account may specify the preferred standard for submitting a purchase order is cXML**
- **order processing: the product order service normalizes all the different conversation languages (say, cXML or ebXML) into a Reuters internal format that can used among back end systems; this translation is bi-directional and (for example) incoming purchase orders in different formats will be translated into a standard Reuters format while outgoing messages already in a Reuters format will be translated into the format suitable for the appropriate Focus Account**
- **order status; this service logs all changes during the entire product purchase life cycle; apart from changes of state, the order status can also host error messages and, should an error occur in a back end system, that error is propagated forward to the order status and logged**
- **IT Asset Management: this uses a digital asset management system to move current product subscriptions from one user to another; for example, an existing subscription for a data feed for a particular exchange could be used by another trader**
- **commercial position: this enables customers to query the status of their current commercial position; this may be done with respect to the organization as a whole ('how many licences for what product and at what price have we bought this year?') but it also assists more targeted queries (such as 'what is the installed position of user X?' or 'how many of product Y do we have?').**

In practice, end users within an organization authenticate against their preferred eProcurement system. Once authenticated, they can start a requisition.

Depending on the accessibility of the relevant catalog, the requisition can span local or remote hosted catalogs. For a remote catalog, the client eProcurement solution must be able to 'punch-out' to the chosen catalog service. For 'punch-out', the eProcurement and catalog services have to conform through a defined protocol for authentication and content transfer. Then the specific catalog service has its shopping cart facility, into which end users add their selections of products and services.

Once the buyer finishes with a requisition, the purchase order is compiled and submitted to the supplier. The purchase processing service authenticates the purchase order against the submitting organization and looks up the organization's collaboration profile.

It is the collaboration profile which accommodates all the collaboration mechanics — such as purchase order structure and the commerce protocols that should be applied. The collaboration profile, therefore:

- **drives the purchase processing service**
- **normalizes all possible inbound message types into an internal canonical form**
- **decomposes the order into smaller processable units that it can be processed by (back end) micro Web Services.**

Once decomposed back end systems will perform further:

- **checking on the user**
- **updating CRM logs**
- **preparing for the service fulfillment.**

An installed position in the back end system hosts all the necessary data. This can be requested by an asset register. Focus Accounts, for example, can synchronize their asset management systems and reconcile their bills.

Products used

The Reuters BPA experimental architecture was built by selecting and deploying what we decided, for our purposes, were the best of breed products. For example, we chose:

- **Oracle's i-Procurement**
- **Requisite Technologies' e-Merge and BugsEye**
- **Bowstreet's Business Web Factory**
- **Siebel's CRM.**

We chose i-Procurement as our procurement system

because of existing in-house expertise and the familiarity of the tool within Reuters for the purchase of resources and stationary. However, our architecture can support any other eProcurement package on the market — such as Ariba or SAP. Another attraction was that Oracle supports the 'punch-out' protocol: this enables buyers to choose products from remotely hosted catalogs.

For the catalog service, e-Merge and BugsEye were chosen as being best-of-breed in the catalog service space. Requisite is a leading catalog software provider, having gained in-depth understanding of catalog related issues from many projects in different domains.

For assembling the business Web Services we used Business Web Factory. Bowstreet is one of the pioneers of Web Services and we saw it as an innovator for Web Service related technologies. Its main product, the Business Web Factory, is being used for the provisioning of our business Web Services which are profiled to deliver a personalized view of Reuters commerce service. The personalization will be achieved by Bowstreet's parametric modeling capabilities, which accommodates the variability side of the personalization. The business Web Services mediate between the micro level back end Web Services.

For the customer relationship aspect we use Siebel's CRM. The latest version of Siebel7 was chosen due to its XML and Web Service support. Siebel will host the user, organization, catalog, product definition and commercial positions.

Recommendations

One of the objectives of Reuters Business Process Automation project was to find out more about the scope of the issues, and then to draw up recommendations on how we should tackle the challenge in the long term. From our six month exercise so far we gained a considerable insight into the shape of the challenges — so much so that we are confident when we make the following assertions and recommendations.

Organizations which are becoming involved in BPA-like activities should, at the least, put the following items on their agenda:

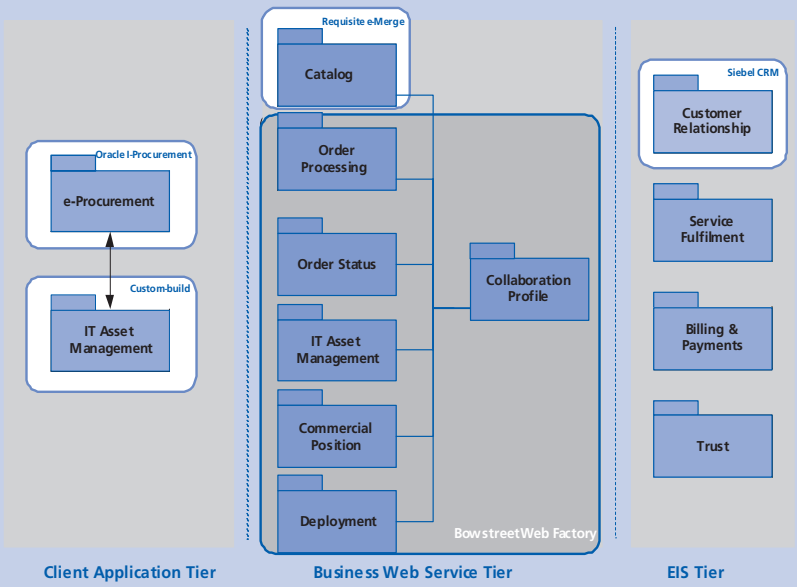
- **product structure and classification**
- **catalog solutions**
- **'configurationware'**
- **vendor evaluation**
- **business Web Services.**

In the product structure and classification arena, organizations planning to apply business process automation must investigate their product architecture. This means they must obtain an understanding of both:

- **the products and services which their organization sells**
- **the marketplaces (and customers) who buy.**

From our experience, such a product architecture can be:

Figure 2.3: BPA high level archeitecture



- **either flat, when it is applicable for a small number of products; a flat approach does not scale because of manageability constraints and increasing the number of products implies data redundancy and additional management**
- **or hierarchical, where the product structure provides more flexibility, dependability and constraints can be better expressed; classification of assets using standard coding schemes — such as UNSPC or eClass — when coupled with the ability to translate between them, further facilitates integration of catalog and search facilities.**

General speaking, for catalog solutions, most catalog vendors support one of these two. However, both approaches frequently lack integration with the other systems that organizations possess.

This means that integration is essential for additional checking with back end systems, for example for product entitlement and permissioning. We recommend you plan to do this integration yourself.

'Configurationware' is our term for exploiting commercial components-off-the-shelf (COTS) solutions. This is sometimes controversial. For example, some COTS solutions provide a closed architecture and come with non-standard interfaces. Others are designed to be open, but the granularity of their interfaces is too fine grain, which results in much additional (and unwanted) coding.

Within the scope of the Reuters BPA project, we recommended that future activities apply the principles of 'configurationware' — using software components or COTS solutions with tuning being delivered in a parameter-driven fashion. This approach appears to pay off — especially for the dynamic assembly of Web Services.

Vendor evaluation is important. On first glance most software vendors praise their product offerings to the skies and claim they can meet all your requirements with ease. The moment of truth comes when the software is finally set up in a lab. It should come as no surprise (yet it almost invariably does) that the actual functionality is either not that expected or is far more rudimentary than the sales discussions led you to believe.

Business Web Services are much in vogue today. In our experience they are a step forward towards providing standardization for integration.

However, on a higher-level, business Web Services remain primitive. This means they are hard to use. We had to undertake the integration that we had hoped (and hope for the future) that others would provide to help us pull together our existing business and technology landscape. Expect to do much more work here than all your vendor discussions might lead you to believe.

Lessons learned

Linking Reuters and customers business systems and processes remains a tough objective against which to deliver. Most systems today are simply not designed for inter-operability. Interfaces, integration points and protocols are often both either too proprietary and/or too rudimentary. To overcome this, vendors and users must develop standard ways at all levels — business and technical — of using technology.

Including variability as a consideration from day one is pivotal if you are to have any hope of supporting multiple business and technical relationships. If you do not accommodate this, organizations (customers and suppliers) will end up with just another point solution — which will produce exorbitant integration costs. Variability has to be built into the technical capabilities in order to ensure that any BPA solution can evolve.

Today's XML vocabularies are an improvement on previous ones. But, at the same time, they can only go so far (which is not far enough). There is still a lot of manual generation, integration, processing and validation required in order to achieve a given level of automation. We have had to accept that XML addresses only structure. It does not provide any semantic or process information.

Lip service continues to be paid when it comes to vendor selection of COTS-based systems, especially in the product catalog space. Most vendors still do not provide rich support for more sophisticated catalogue requirements. This meant more work for us (and will for you).

Finally, running an experiment as a pre-project certainly helped us to understand the practicalities of the BPA problem space. Technical delivery teams are often too pressured on release dates and so do not have the time to work out a sound strategic solution. Undertaking a pre-project offers the time to think and to experiment. The time taken will be more than compensated for when true BPA is commenced.

Management conclusion

Business Process Automation can often be made to sound as if it is merely the collection together, and integration of,

selected existing applications or parts of applications. To start from this viewpoint is, however, to:

- ***constrain what will be delivered***
- ***underestimate the effort required.***

In this analysis, Mr. Bettag and Mr. Curtis have gone much further. The BPA which Reuters seeks has to do with making it easier to simplify (and enrich) the customer's experience of doing business with Reuters while also providing a means to reduce Reuters own internal processing costs. The requirement, therefore, spans both intra-company and inter-company processing — the latter being the most difficult of all BPA to deliver.

As Messrs. Bettag and Curtis illustrate, the path forwards is not yet simple. Much more needs to be done by the purchaser of BPA than you might wish. In particular, expect integration — including Web Service integration — to be an internal activity, not something you buy off the shelf or expect to obtain from a vendor.

The Eighth Layer: it is all about middleware

Peter A. Bernstein
President, Infonautics Consulting

Management introduction

As information technology (IT) industries continue to struggle through what is (now) two years of digital discontent, it has become apparent that the leaders of Technocracy have lost sight of what is happening and what is valuable. Three things are becoming ever clearer. We are:

- ***not living through a 'technology revolution'***
- ***rather we are living through a 'value revolution' that is being enabled by the acceleration of technological inventiveness in a world where there is no longer a monopoly on innovation***
- ***inexorably moving more of our lives online; as this occurs value will be created — and winners and losers determined — at the intersection of the physical and virtual worlds.***

This intersection is what Peter Bernstein calls 'The Eighth Layer'. As described in this analysis, the Eighth Layer has many constituent attributes — places, sets of processes and states. That said, the Eighth Layer's principle attribute is that it is the logical extension of networking's OSI model — for it encompasses a dynamic, organic, interactive relationship mediation capability.

In plain English, it is the ultimate definition of 'middleware'. It straddles the interface between the physical and virtual worlds — and creates benefits for both.

The Eighth Layer is middleware — the intersection of the physical and virtual worlds

Robert Metcalf, the founder of local area network innovator 3Com, is credited with coining a fundamental law of the communications industry. Metcalf's Law says that the value of a network increases by orders of magnitude for each additional connection that is made to that network. Although this can be expressed mathematically in several different ways, my favorite is the simplest: 2+2 > 4.

Metcalf's Law is as important a concept to the communications industry as Moore's Law is to understanding the historic speed of innovation in the computer industry. The former encapsulates the essence of the value of interactive communications networks — from the invention of the telephone to the popularity of the Internet (that started in roughly 1996, with the commercialization of the Mosaic Web browser).

The history of value in the telecommunications industry is both a chronicle of the continuing journey and the metamorphosis of technology-enabled paradigm shifts. It is an excellent starting point for establishing why the Eighth Layer matters.

The telco experience

In the telco world, the original business model was based on the notion of customers paying a monopoly service provider a regulated rate-of-return (at a significant price) in order to be connected to everyone else. This was deemed to be a 'good thing'. A vertically integrated monopoly industry structure emerged, where the same telco entity owned:

- **the infrastructure**
- **the services that ran over it**
- **the means to provide for its improvement.**

This business model continued to expand. It moved from local support into a global industry dominated by a few national champions (often arms of the local postal regimes). But, in the early 1970s, competition began to appear — starting in the long-distance and customer premise equipment (CPE) industries.

Over time, and aided by the likes of MCI, the vertical monopolies, like AT&T and other traditional telecommunications companies, became victims of 'disintermediation.' This is the fancy term for the systematic deconstruction of big markets by highly focused competitors who offer (and deliver) niche functions far better than incumbent vertical giants operating across broad markets.

Disintermediation was not specific to the telco world. It has also occurred in the financial and energy markets, to name but two others. While some prefer the term 'horizontalization', I will describe what it delivered.

Three concepts underpin disintermediation, namely that:

- **choice is good**
- **buyers will purchase rationally**
- **buyers will make the effort, when offered choices, to piece together their own optimized solutions (based on their perceptions of what is valuable to their own organizations).**

In the second phase of the telco industry's evolution — lasting from 1971 through 1999 — enormous wealth was created by companies operating in relatively open markets. The option to choose between different long distance carriers or equipment providers meant the era of scarce information (that was locked up in the 'old world' media of paper) that had to be accessed over scarce communications resources, was destroyed with the arrival of personal and corporate computing and the Internet (of which more later).

In fact, the Eighth Layer, and value creation, are the natural consequence of what happens when there is an over-abundance of access, transport and processing. In the telco service area, in particular, competition became extremely dynamic as competitors went after those areas in the value chain where there was most to gain. As MCI proved, the biggest of these was where the long distance meter ran.

Although local telephone calls represented 75% of all traffic, most of this was at a flat rates. In contrast, long distance calls were charged for on the basis of distance and time. The more we talked, and the farther the distance between the calling parties, the more money was charged. Thus the variable revenues associated with premium traffic revenues was the logical target for competitors for the incumbent telco monopolies.

Another way to look at the history of communications is to describe it as the desperate attempt by incumbent vendors to leverage their flat rate businesses and/or to replace this with a metered businesses. This is the reason why the cable TV industry, for example, needs pay services like video-on-demand or telephony. These fund the continuing upgrading of the black hole known as the physical network. Not co-incidentally, it is also explains why the software vendor

community is so enthralled with the prospects for Web Services — applications that ride over physical infrastructure and can be paid for by the session or by the amount of the data being 'dipped'.

Lurking in the weeds

The problem lurking in the weeds has always been the data communications business. This has never been metered. Instead, data communications business billing was, and is now, predicated on speeds, ports or distance between the end points of a connection.

Right around the new millennium, mainly as a result of the explosive growth of first facsimile and then corporate network computing and the Internet, data traffic over telecommunications networks surpassed voice traffic as the information most transported around the block (and around the world). In addition, the mix of traffic — that historically had predominately been local — changed. Digital, packet-based networks which route traffic according to a different kind of networking logic began to dominate (Voice over IP is only the most obvious example).

Unfortunately for the traditional owners of local exchange networks, voice still pays the bills. It is one of the major reasons we are stuck at the present time. The companies which own the voice business have no real incentive to provide a data communications infrastructure that will expedite their own commercial demise.

A second reason we are stuck is because of the meltdown of the profitability of the long distance companies. The variables that impacted on this meltdown — ranging from the Internet, the way in which the wireless industry has grown, the manner in which public policy was created and implemented, the irrationality of financial institutions funding poor business plans which led to the current fiber glut, etc. — need serious crunching in a building full of super computers to determine what exactly went wrong. Nevertheless, the outcome is becoming obvious. A subtle value shift, which occurred in the late 1980s and through all of the 1990s, is boiling to the surface.

Avarice leads to an appreciation of value

Much of the success of the 1980s and 1990s was fueled by avarice. Metcalf's Law was modified and put on steroids. A consensus formed around the idea that the world was not merely about leveraging the growing number of connections but was built on controlling the new and/or improved services that were transported across networks. This was achieved by leveraging the physical network as the services gateway/gate keeper.

Put another way, the consensus was that the only way to ensure customers stayed loyal — and passed through your toll booths — was to own the facilities that attached customers to networks (in effect commandeering Layers 1-3 of the OSI Model). Much of the communications industry merger and acquisition activity from 1988 until now was driven by this — and, arguably, we have yet to let go of this notion. Indeed, we are absorbing the consequent pain.

Back to the history for just a moment. The captains of the telco industry believed that you had to own the facilities in order to own the customer. They set out on a buying spree, one which was augmented in the U.S. by the Congress in 1996 when it encouraged others to build their own competitive facilities.

This started a buying spree that was subsequently aped in the cable industry, and was brought to magnificent craziness by the global wireless spectrum auctions (paying billions for 3G licences in the UK and Germany was the pinnacle of this foolishness). Nobody wanted to believe that value could be created by separating infrastructure from content and context mediation.

The reason was that no one trusted their abilities to attract and keep customers in any other way but the one which was familiar — owning the infrastructure. At the same time, however, there was a tacit realization that the services offered, absent the physical connection, were becoming less valuable in the eyes of the consumer.

We all know what happened. Bankruptcy and class action lawyers found themselves an annuity business. Worse, because of the lack of a clearly defined alternative model for profitability, the knee-jerk reactions of executives were to retrench to past practices, even though these might no longer be applicable. This explains why the pain of the current paradigm shift is lasting much longer than it should.

Where the telcos, and others, have gotten it wrong is that they think they are crossing Geoffrey Moore's 'technology chasm' — and moving onto the next successful part of the 'S' curve. In reality, in my observation, we are moving to a new paradigm where value — and not technology for technology's sake — matters. This is the Eighth Layer. Its foundation is the proper use of middleware to create value.

Enter the Internet

Before examining the Eighth Layer in more detail, the influ-

ence of the Internet needs discussion. With the Internet, Metcalf's Law was transformed. What most observers have missed, and the 'dot.bombs' never understood, is possibly the most profound of the many incredible impacts of the Internet:

- **Yes, the 'freeing' of information — which emboldened entrepreneurs and the technology food chain to package information in new and (hopefully) profitable ways — is huge**
- **Yes, putting choice at the click of a mouse in the hands of consumers is earthshaking**
- **Yes, the ability to produce in the virtual world transactional experiences that cannot be replicated in the physical world is 'disruptive.'**

These will be the lasting legacies of the likes of eBay and Amazon.com. But equally as profound was the shift in value in the interactive communications industries. What the Internet changed was the value chain associated with owning customers and the way these customers physically attach to a network.

If AOL has proved nothing else, it is that in the new Internet world, owning the relationship (not the connection), is what counts. This does not mean that the facilities business is a 'bad' business. What it does mean is that the facilities business will become a commodity business which cannot support a large number of players. Everyone else is going to have to move up the food chain and develop premium capabilities for which they can charge premium prices.

What AOL has (that Yahoo! does not) are customers who find being in the AOL 'E'-vironment so compelling that they will pay an annuity of $21.95 per month to be connected. In fact, it is more than this. They are paying to be members of a community — an 'E'-vironment — that is accessible from anywhere over any media. Keeping this locale desirable — and adding to it so that the customer's willing commitment will rise to (say) $200 per month — is AOL's real challenge.

Here the kick obtained from Metcalf's Law, combined with the bundling of enhanced communications and interactive entertainment (video, music and gaming), applies. The proof of the law of big numbers — that Metcalf customized for communication and networking — can be seen in AOL's Instant Messenger. This proves that the value of a network is some multiple of the number of individuals who are connected to a compelling experience.

What the long distance telco companies are discovering (all too slowly) is that they were not offering compelling value. Merely possessing a physical connection and providing a bill to the customer is not sufficient.

The important caveat going forward comes with the transfer of customer control primacy from the physical to the virtual connection. It is all about the customer experience, because the experience dictates the longevity of the relationship. The value-add of the 'new' networks exists in their mediation of content and context — the Eighth Layer. This resides in the:

- **ability cleverly to monopolize the one scarce resource we cannot create more of: time**
- **recognition, in an era dominated by a cornucopia of information-based distractions, that being the gate keeper/gateway between Virtual.ME (V.ME, see below) and the tangible world is the sweet spot of the next information age.**

The value in the future of networking is not just about V.ME being connected to the rest of the world safely and securely. It is about:

- **the organic processes embodied in middleware that safely filter the world for ME**
- **creating order from chaos**
- **giving permission, and authenticating the bit streams aimed in my direction (so I can process them according to my needs)**
- **providing verification of my abilities to transact things when I am the bit stream aimed at somebody else's 'E'-vironment.**

This is true for both individuals and organizations. Companies that do not understand this are doomed to failure. They will have failed to grasp that, for the foreseeable future, they are operating in a buyers' market.

What is the Eighth Layer?

The Eighth Layer expresses — or captures — the value which the OSI model was supposed, but never quite managed, to deliver. It is multi-functional and is best described as a collection of:

- **virtual places**
- **sets of processes**
- **states.**

Virtual places cover the intersection of the virtual and physical worlds. They are where the virtual embodiment of my personal volition (or choice) — V.ME — resides. This, in

reality, includes multiple MEs in terms of my various electronic personae. But it can include multiple contexts, for example where to interact with an individual, work group, enterprise, community, etc.

It is not just virtual because it resides in the ether. It is virtual because it is a collection of high speed interactions that is nowhere and everywhere at the same time. It is a place by virtue of the fact that it is my designated destination for those who wish to interact with ME.

In contrast, sets of processes are the palpable expression of how my electronic volition interacts with the rest of the electronic world. This is defined by a set of processes — the policies and rules by which it (my volition) engages the electronic impulses of others — and they in turn in engage me, and everyone and everything that wants part of my time.

The processes that define the various contexts of this V.ME are not to be confused with their first generation cousin, online digital assistants. When you hit V.ME, you are not reaching my assistant, you are reaching the electronic embodiment of ME as a person. As such, V.ME can be any, or all, of the personae just listed depending on what I am doing and as part of whatever other grouping (including multi-processing) in which I am functioning as a member at any point in time:

- **some call this physical/virtual intersection names like 'Portal' or 'Organic Persona'**
- **Microsoft calls it '.NET My Services'**
- **others classify this as the 'personalization' part of 'Web Services'.**

On the other hand, we do know that whatever the name, such physical/virtual intersections will be software derived and defined 'E'-vironments.

In this context, state encompasses the dynamic relational and transactional state. My V.ME will always be:

- **on**
- **interacting**
- **ready to be reprogrammed (or deprogrammed)**
- **synchronized**
- **artificially intelligent.**

It will take what it knows and learns. It will add value to its capability to act as my proxy. In effect, the Eighth Layer is 'technology with intent.' It is the means for establishing the compelling experiences we all crave — the icing on the engineers' cake.

The Eighth Layer is a relationship layer

What are the characteristics of the Eighth Layer? The Eighth Layer needs to be viewed in the context of the marketplace it is facilitating. This means different things to different people depending upon whether its role is being defined in an organizational or personal frame of reference. (Please notice I am being careful not to employ commonly used terms like 'business' or 'enterprise'. This may seem like semantics, but there is no need to exclude from this discussion governments, schools, not-for-profit and other organizations — which do not think of themselves as businesses or enterprises — even though the hype machines of many IT vendors seem to prefer to do so).

The Eighth Layer also means different things depending upon its use in describing a place, process or state. In an organizational context, one must start with why managers with IT purchasing and management responsibilities buy IT in the first place — and communications networking products and services specifically. This is fundamental to understanding the Eighth Layer. It is the combination of place, processes and state.

All organizations, regardless of their size, buy computers and communications products and services in the hopes that these will provide answers to many questions, such as:

- **how can I, on behalf of my organization, obtain optimal performance from my network to achieve the organization's business objectives in the best way possible?**
- **how can I be sure that the end results are compelling, for example, that they satisfy or exceed desired outcome criteria?**
- **how can I satisfy the needs, and consistently strive to exceed the expectations, of those who turn to me and my organization for IT solutions to their problems?**
- **who are the best vendors to take my/our IT infrastructure to where it must be while ensuring maximum asset performance, easy accommodation to dynamic changes and minimal impact on budgets, staff and end users?**

All these questions ultimately relate to people, performance and relationships (trust) — and the tools used to bring or add value to each. Technology in all these settings must 'work.' This means not just that it should function flawlessly. It means providing appreciable and measurable results that can be construed as having value.

Specifically, IT in most organizations — if it is to be viewed

as providing value — must please customers, whether internal or external. This may be achieved by (for example):

- **increasing productivity by helping people manage their time better or by shortening product and service development cycles**
- **increasing revenues by speeding transaction times, by readily accommodating higher transaction volumes, by pleasing existing customers as well as easing the processes for obtaining and retaining new customers (see also page 12, and Reuters thinking on this)**
- **cutting costs for the creation and delivery of products and services, the operation and maintenance of the infrastructure components while being ready for potential expansion**
- **solving as yet unresolved business issues**
- **discovering and then creating new, possibly differentiated, business opportunities that are profitable.**

At the heart of all of this is the creation of a trusted relationship between the stewards of technology resources in organizations going simultaneously in multiple directions and their vendors. For instance, these IT stewards must have confidence that their vendors have:

- **a vision of the future that possesses credibility — not just a technology blueprint but also a clear view of the roles and responsibilities of customers and suppliers**
- **the intellectual and financial resources to work with customers to execute against agreed-upon plans and actions**
- **the willingness to respond quickly and effectively to changing circumstances.**

In short, what IT stewards desire is a strategic supplier who is willing to, and capable of, sharing risks as well as the rewards. To achieve this they must have the backing of their management plus the assurance that the business case for new product or service acquisition and deployment rests on solid grounds.

This means the deliverables must have a reasonable expectation of producing the benefits that have been promised. It means metrics must be in place in advance so that it is possible to assess the progress toward the intended benefits. Furthermore, there must be proof that the IT stewards have pleased their users, and that those users are willing to continue to trust these stewards to work in deploying technologies that operate in the best interests of the users with a maximum of performance and a minimum of pain.

In short, IT asset managers must have trusted relationships with IT vendors that ensure their products and services have the highest levels of:

- **programmability (a combination of flexibility and ease-of-use)**
- **reliability**
- **manageability**
- **accountability.**

Trust and the Eighth Layer

If I am an IT asset manager in The Eighth Layer world, I need to ensure that my organization's:

- **place is always available and always secure**
- **processes are private — according to my instructions — and delivering the results that are expected when users visit**
- **state changes can be easily accomplished, and verified, so that people can be held accountable.**

In short, I need to trust that my network is operating at peak performance around the clock and that my middleware is doing what it is supposed to be doing on my behalf — mediating my volition with that of the rest of the online world that desires to interact with me or on my behalf.

In a personal context, the differences between our organizational needs and our personal ones are quite subtle. We all want technology to be working for us rather than against us. However, we actually want even more security and privacy (Big Brothers need not apply, unless it is a government on a national security mission). We want ever more programmability — because we need technology to act as our:

- **window on the world**
- **agent in it, and to it**
- **intelligent filter from it.**

At the same time, we need technology to do this without such involvement becoming a burden on our time (or bank accounts). We need it to protect our precious cargo from those who might steal it, yet be capable of exposing itself in times of national crises (so that the greater good is not thwarted).

Take the number of entities that attempt to reach you

today at work online. Multiply that by the total number of entities with access to way too much psycho-graphic and demographic information about you (who can target you as a potential customer, or as a victim for a malevolent act), and you will begin to understand why personal needs may exceed my business ones.

Thus, in a personal context, the middleware — for it is this that delivers V.ME — needs always to be vigilant, constantly reactive and still adaptive to changes in my context and volition as I move about in my .ME world. Put another way, middleware needs to deliver what I call 'context mediation.'

The Eighth Layer is a context mediation layer

Context mediation is not a new spiritual movement. It is nothing more complex than acknowledging or reflecting change — in the place, processes and state — that comprise the Eighth Layer and make on-the-fly adjustments to my policies and rules.

It must organically, competently, reliably and securely relate to 'E'-verything. This means it must work with all of the volition embodied in the 'E' proxies and agents (content) sent its way, or that it encounters when I send it out to work magic on my behalf.

At the heart of this capability must be bullet proof authentication. For the Eighth Layer to function and be useful, I need to know with absolute assurance that you are who you say you are when we interact remotely. In return, you are going to want to want to know absolutely that I am who I say I am, and that I have the resources and authority to interact with you according to the policies and rules you have defined as your baseline for engaging in our interaction.

The Eighth Layer as context mediator even has it own mantra: 'everywhere, every time, all the time, over all media and at the right time'. Whenever you need ME, I am there — because of middleware.

Why middleware is the key

In this context, it is middleware which creates the place, processes and states of the Eighth Layer. Hence, understanding the deliverables, and myriad, of new requirements of the next generation of middleware as the electronic proxy for my volition, is the only way for vendors to be successful.

As stated earlier, the recognition of this reality — and changing selling behavior adapted to the new dynamics engendered by the emerging era of user control — is not going to be easy. We are at a major inflection point in the history of IT. The fortunes of vendors are going to be set for decades in the next several dozen months.

What customers and vendors alike should understand is that it is middleware which is delivering the disruption of the historic relationship between technology sellers and buyers. Middleware has become an agent of change.

It is one of the great ironies of the Information Age that those who have preached the mantra that the only way to succeed is to make change your friend and flexibility your corporate culture, are the ones having such a difficult time adjusting to the world which they have wrought.

In the Eighth Layer world flexibility is fast approaching. It is likely to become pervasive, accommodating, enhancing and creative. It will leverage middleware so that the latter becomes the gateway and gate keeper to the transactional ME — the path to economic vitality.

Management conclusion

In this examination of past events and likely trends, Mr. Bernstein examines what is going to be valuable to society in the future. Unlike the past, the vendors of technology solutions will diminish in importance. Instead, what will matter is the ability to exploit the Eighth Layer — and V.ME. The next generation of wealth will be determined by what we (individuals and organizations) do and how we react. It will, less and less, be about new technologies aspiring to dominate markets.

Instead, the new focus will be on trying to monopolize 'my time' — interacting with my world safely, and securely. If you can do this you cannot help but monopolize my transactions. Hence the importance of middleware.

Those who do not understand this new value paradigm — and remain convinced that building better technologies than the next guy is the key to success — will fail. Instead progress will be delivered by those who understand how creatively to leverage middleware to deliver improved value, time and place.

Web Services — for whom are they relevant?

Mark Lillycrop
Consultant

Management introduction

It is very easy for observers outside the IT industry to see new trends and technologies as hapless solutions in search of problems, appearing from nowhere, absorbing vast amounts of precious investment and vanishing as quickly as they had arrived. Financial directors are particularly prone to this view, and 'the case of the disappearing technology' has often been responsible for the antipathy between CFO and CIO. With product marketers leaping hysterically from one industry buzzword to the next, it really is easy to lose track of the business justification for many new tools and products.

The latest enthusiasm of the vendor community is Web Services. And, of all the Internet technologies that have tumbled onto the market in recent years, Web Services seem particularly hard pressed to find a business need to satisfy.

Indeed, nearly all leading middleware vendors seem intent on placing their futures on a bet on the Web Services sector, even though only a few — most notably IBM, BEA, Sun, Microsoft and Oracle — are likely to have any real impact on their future shape and implementation.

In this analysis, Mark Lillycrop examines two important questions:

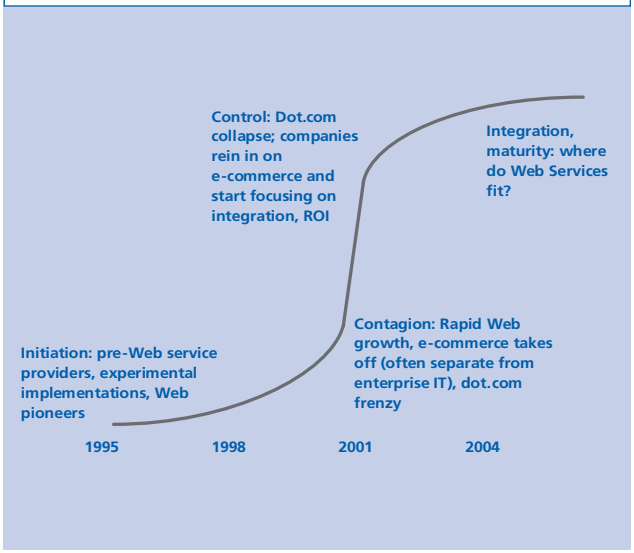
- ***do Web Services themselves have a serious role to play in business, or are they just an interim stage on the way to a new generation of Web-based applications?***
- ***if this role is a serious one, what specific criteria do Web Services need to satisfy?***

The challenges facing Web Services

To understand where Web Services have come from, it helps to plot the growth of e-business on a Nolan Curve. Dick Nolan's S-shaped graph (Figure 4.1) has been used by generations of management consultants to follow the development of new technologies:

- **from slow take-off at 'Initiation'**
- **through the rapid growth, vendor hype, and user acceptance of 'Contagion'**
- **to 'Control', where growth slows and management becomes the number one priority.**

Figure 4.1: Nolan Curve



Thereafter, each technology heads steadily towards integration with other elements of the IT architecture. Finally it migrates into technical maturity.

If we think of the e-commerce revolution (and I do not think 'revolution' is too strong a word), we can create a Nolan curve that looks something like Figure 4.1. We have moved from the early (and relatively slow) development of Web and pre-Web technologies in the early to mid-1990s, through the consumer- and VC-driven insanity of the dot-com boom, to the Control phase — which, for many large organizations took place rather suddenly and dramatically last year. It was at this point that:

- **many companies took stock, pulled in the reins and considered just what business benefit they were obtaining from e-commerce**
- **EAI came to the fore again, as e-business solutions that had been hurriedly bolted onto the IT architecture were re-engineered and integrated to existing back end applications**
- **vendors, most of whom had cheerily oversold software and hardware capacity during the rapid growth phase of e-commerce, realized that their revenues were about to dry up.**

While all this was going on:

- **Microsoft was assembling its grand plans for Web-based desktop domination — in the shape of .NET**
- **IBM was searching for a coherent way to bring together its disparate Web server, messaging and transaction management products under one brand — WebSphere.**

Both of these players, plus a large number of their competitors, came to see Web Services as a convenience by which they might define the next stage of e-business, as it edged its way past Control and headed towards Integration and Maturity. The problem was that, in the Control phase, a number of key technical issues had been identified that had yet to be addressed if e-commerce was to continue to contribute to new business processes (and, even more importantly, to new business). These included:

- **extending the enterprise**
- **mobilizing the client**
- **keeping the enterprise secure**
- **providing performance and availability**
- **delivering scalability.**

Extending the enterprise

During the contagious phase of e-business, companies at the forefront of Internet exploitation began to change shape. We entered the age of the 'extended enterprise', where traditional boundaries — especially those between the data and IT resources of one company and those of its business partners and/or customers — gradually broke down.

Admittedly this problem had been around for a long time. Mergers and acquisitions almost always require a major re-engineering exercise as disparate hardware, software and data types are brought together.

What changed, though, was that the Internet introduced an expectation of a far greater degree of flexibility than had been the case before. Today's electronic supply chains and

extranets involve 'virtual organizations'. The integration of resources is occurring not only within one organization but between many third party organizations. This demands standards and rule-sets to accommodate such changing relationships.

Mobilizing the client

The biggest growth area of e-business is now the mobile sector — characterized by hand-held computers and PDAs, cell-phones, and a plethora of 'pervasive' devices that are redefining the way users interact with business applications. Worldwide cellphone subscriptions are tipped to hit one billion this year, and they are growing much faster in many parts of the developing world (where they are providing reliable communications for the first time) than in bandwidth-constrained Western countries.

As a separate island of technology they are fine. But, in an environment where the user interface once came down to a simple choice between 3270 or 5250 or VT100 terminals, the range of mobile client devices poses yet another integration problem for IT.

Keeping the enterprise secure

Never has IT security 'enjoyed' such a high profile as today. The Web has pushed personal privacy, hacking and cybercrime to the top of the agenda. Seemingly every new development in Internet functionality opens up new areas of exposure.

Microsoft is so concerned about its reputation in this area that it has introduced its ***Trustworthy Computing*** initiative, and given its internal security inspectors greater powers. At the same time, Oracle's 8i and 9i database/application server users are being warned of 37 possible exposures.

Mobile applications, moreover, are notoriously insecure. In addition, virtual organizations (as discussed earlier) introduce problems of their own.

Netted out, the problem for most organizations, governmental or commercial, is how to devise workable security policies for:

- **staff (over whom they have some control)**
- **business partners (and their staff), over whom there is much less, or no, control**

at a time when the distinction between employees and outsiders is in a constant state of flux.

Performance and availability

Traditionally, enterprise systems are fine-tuned to deliver ultra-high levels of availability. Each successive release of operating system, database, transactional software and application is designed to raise that level even further.

With Web-based applications, where the enterprise does not necessarily control (or even monitor) end to end performance, this virtuous improvement cycle may not be able to exist. There is a very real danger that anything that is an application which is Web or Internet -based may compromise expected levels of IT service delivery.

Scalability

If e-commerce changed one thing for good, it was the concept of capacity planning. Suddenly, with 24x7 applications available through the Web and subject to hopelessly unpredictable usage 'spikes', it became clear that the only practical Web solutions were those that could scale almost exponentially. In many areas, the spikes became less problematic after the dot.com furore died down.

Nevertheless, the lessons had been learnt. With international Web-based services, scalability is everything. Just ask Victoria's Secrets about the SuperBowl or the International Olympic Committee about the Olympic Games.

Development is where you start

Despite their operational connotations, all of the above issues start with the development environment. Whatever comes next in the way of Web Services and middleware, and whoever provides them, there is a manifest need to address all of the above requirements — flexibility, performance, security, ease of use and implementation and, above all, scalability — as well as delivering new function to developers and users.

But there is more. Vendors need to decide precisely who the target user of these new services really is. Raw development tools may be fine for professional vendor developers working on specialty Web applications of their own. To achieve success and reach any broader market, Web Services need not only to embrace the principles listed above but must be packaged and usable in ways that make them relevant and appealing to less technically enthusiastic customers.

Where are the middleware/ Web Services vendors today?

By making the point that Web Services, as middleware,

must transcend systems programmers and their like I am only drawing attention to an unpleasant reality — that this is where middleware vendors continue to struggle. As is all too common, the leading protagonists are interpreting the aims and targets of Web Services in their own different ways. There is little real progress — as measured by what the normal business person might achieve or expect to achieve (with Web Services) — despite the endless hype.

A swift assessment of the Web Services market place today highlights the problems:

- **Sun still keenly champions the Java cause while simultaneously wrestling to deliver the latest version of Java 2 Enterprise Edition and blaming its slow progress on the SOAP (Simple Object Access Protocol) definition; little improves or changes here — with Sun preferring to criticize rather than contribute or deliver**
- **BEA, with WebLogic and associated products, is claiming solid market share — in part due to its successful support of the second version of UDDI (Universal Description Discovery and Integration); but BEA is increasingly being confined to a niche market (application servers) and is becoming heavily reliant on Sun's Solaris platform even though it is trying valiantly to diversify and expand into the Intel world**
- **Microsoft is, very slowly, rolling out products that adhere to its .NET strategy and has a reasonably healthy developer program underway; nevertheless, concerns over the security implications of HailStorm and the Passport initiative, plus Microsoft's unique and rigid interpretation of Web Services in general, leave many question marks hanging over the enterprise applicability of the .NET approach beyond Windows**
- **XML, one of the core technologies of Web services, is in serious danger of drowning in its own complexity; already it possesses endless sub-standards, with more and more schemas emerging from various bodies, while the W3C battles to maintain some form of control.**

Finally there is IBM. It has a vast repertoire of middleware and intellectual resources. It is equipped to provide not just the tools to support all-singing, all-dancing Web Services but also the services to support them (so long as you have a deep enough pocket to pay for (IBM's) Global Services to master IBM's arcane product delivery).

Yet IBM continues to wrestle with more focused competitors for market share. Despite its familiarity with the needs of large-scale transactional systems, IBM still has a great deal of work to do before its WebSphere product umbrella progresses beyond name-level integration of its diverse TP, messaging, database and application server product sets, to deliver a fully integrated Web Service delivery platform with any semblance of friendliness and usability by ordinary mortals.

Who are Web Services for?

The issue of IBM's Global Services organization brings us back to the question of who is the target user base for Web Services. After all, much of IBM's continued success in recent years has been attributable to the way it has moved products and tools away from its customers, and replaced them with service offerings.

The signs are that IBM sees Web Services are the next big thing for its consultants and support operations. But, outside the small core of very large organizations with which IBM is so familiar, such services will only succeed if customers:

- **understand the potential of Web Services**
- **know how to exploit them**
- **can assess the return on any Web Services investment within the enterprise.**

Since the dot.com fall, few major IT decisions are being made without close attention to ROI. Thus far the leading middleware and Web Services vendors are failing to communicate what are the benefits of Web Services to an organization.

The skills problem

If Web Services should have one altruistic aim in mind, it should be to help alleviate the constant IT skills crisis. Most organizations of any size are looking for ways to build upon their established Web applications, and to provide new benefits in terms of customer interaction. Most, though, are facing a shortage of appropriate skills.

This does not necessarily come down to difficulties in recruiting programmers with experience of the right tools. Sure, specialized knowledge of Java, XML, SOAP, etc. are highly prized — and the best people will gravitate towards the highest bidder. But, on the whole, good programmers with out of date skills can be retrained (indeed, regular training is the best way to make sure that highly valued staff do not go looking for alternative employment).

The skills which organizations really lack lie in the areas of integration and project management of integration. As organizations focus on drawing together their initial Web projects, integrating them more closely with back end systems, and exploiting the opportunities afforded by the 'extended enterprise', matters more and more. This only emphasizes that the people who really matter are those who understand the culture of an organization and the way its business processes work together.

These people could come from a more traditional CICS/COBOL background. Equally, they may come from a non-technical background, like a line of business. They may have knowledge only of high level development or database tools. While these highly scarce integration specialists might be looking for Web Services to complement existing skills, this will only apply if Web Services:

- **require minimal technical knowledge**
- **address the management challenges these people expect to solve.**

Unfortunately, looking at today's Web Services offerings, there is little sign that products which possess such attributes will be forthcoming — although the one exception may be Microsoft. It knows about usability issues and making tools or approaches attractive. On the other hand, it still shows little understanding of the back end management issues that are so important a consideration in integration (all the world is not a Windows-equipped desktop or server).

At the other extreme, there is no doubt that IBM would like to package WebSphere-based services in a form that is acceptable to the small and medium business market. This is an admirable objective but it is not one where IBM generally shows finesse. It needs to avoid its usual tendency of offering a hydraulic-powered, steam driven sledgehammer to crush a grape.

Management conclusion?

In many ways, it seems, Web Services are a half-way stage towards a new mechanism for delivering Web-based applications, or more likely several such mechanisms.

Once the technologies underlying Web Services have reached reasonable maturity — and common standards have evolved to embrace the management requirements outlined above — we are likely to see competitive vendors focus on different parts of the market, for example:

- ***IBM concentrating on customized tools and service-based offerings for large customers***
- ***Microsoft (and its development partners) going for a packaged approach for the (numerically) broadest customer base***
- ***a selection of others vendors specializing in specific sectors or segments (for example, mobile and telecommuting).***

Web Services currently present a confusing picture, characterized by over-taxed standards bodies and a group of heavyweight vendors (IBM, Sun, Oracle and Microsoft) with conflicting views about the future. It would be nice to think that, over the next few months, some of these differences will be settled and that we — the customers — can sit back and watch the e-commerce Nolan Curve complete its journey to integration and maturity.

However delightful such a prospect is, Mr. Lillycrop is pessimistic. The main protagonists seem to have forgotten that making a product or service tangible is often a prequisite for a sale. Web Services, even more than most middleware, are elusive, intangible and contradictory.

While this persists, customers will find it extremely hard to find the justification to buy into what all too often looks like a vendor dominated vision for simplifying their own — rather then their customers' — problems.

31

On the declaration of integration: the declarative vs. procedural

David McGoveran
President, Alternative Technologies

Management introduction

There exists today the notion of a 'business analyst' — a non-IT person who can use IT constructs to make his organization more flexible. Such individuals would be immensely useful, if you could readily find them.

The reason that they are so rare is bound up in a complex and, at times, overly archane argument that is usually labeled as the declarative vs. the procedural. But the cause of the discussion is important. It is about enabling business-oriented, non-IT trained people to create business processes from existing IT constructs. The potential value that can be unlocked is enormous.

In this analysis, David McGoveran, the President of Alternative Technologies, of Felton, CA:

- ***illustrates why the procedural approach, so common to IT, has distinct problems***
- ***shows why the declarative approach has both more appeal and relevance***
- ***relates both to the ever more important world of Enterprise Application Integration (EAI) and the need to increase inter-application capabilities at the business level.***

The procedural problem

I am going to start by examining the procedural before moving on to discuss the declarative and why the world of EAI needs the latter rather than the former.

Over the years, many techniques and technologies have been used to address the challenge of making computers do what people would like them to do, with varying degrees of success. Unfortunately, although a particular language and development environment may reduce some of the challenge, there are inherent problems with procedural computer languages.

By a procedural language, I mean one which requires a developer to specify how to accomplish a task and with what resources. In this context, procedural languages have at least one of:

- **procedural constructs (such as conditions and control loops)**
- **physical resource specifications (such as structured data definitions)**
- **procedure definition and invocation capabilities.**

As much as traditional IT developers love to design and develop software systems, and as much as our employers may enjoy paying for them, we can no longer afford to integrate enterprise applications through the use of procedural languages. To understand why, it is necessary to step back in time and examine why the procedural approach remains insufficient.

Integration before and after 1995

Enterprise Application Integration (EAI) experienced something of a revolution in the years after 1995. Prior to then, almost all application integration efforts were one-off, point to point and highly custom efforts.

As you might expect, these were also costly, with intense use of specialty skills. Perhaps only those who worked on such projects can recall just how much effort was required to maintain such application integration. Working without the benefits of a common infrastructure (standard interfaces, integration brokers, and so on), applying new or additional function changes to applications all too easily damaged any previously working integration.

Over time, through the gradual introduction of standards and shared integration components, levels of abstraction were raised. With this it became possible to consider strategies for larger scale integration efforts.

While we may debate the relative benefits of various integration architectures (point to point, hub, multi-hub, distributed, etc.), each was significant because it separated the means of accomplishing a goal from the goal itself. By creating integration services (messaging, data transformation, routing, etc.) with well-defined interfaces, we began to obtain the benefits of architectural abstraction.

Arguably, it is this encapsulation of generalized and reusable functionality that made the EAI revolution possible. It also underlines why the procedural approach remains deficient.

Business requirements keep moving

As many have discovered, there are further methods by which we can improve upon this abstraction. For example, rule-driven methods can be used to change the behavior of integration services. Even more attractive is the fact that rule changes can be made while systems are online. This greatly lowers deployment time and costs — as well as offering tremendous flexibility. In addition, rule-driven routing enables implementation and control of message flows — also without requiring shutdown, programming, or re-deployment.

As a second example, process model-driven integration — as found in a BPMS (business process management system) — takes such capabilities to higher levels of abstraction, by enabling the flow of business events and activities to be described and implemented graphically. Although few BPMSs are yet so sophisticated, it is now conceivable for someone with no knowledge of computing to define and implement business processes.

With every increase in the level of component abstraction and each move toward a distributed services architecture, the ability to deliver a function with fewer design and development errors has improved. This is not merely a consequence of simple component re-use. It is also one of making the relationships among components simpler so that they can be combined more reliably.

The contrast here is with the procedural approach to application integration. This was achieved only with much detailed work using application programming interfaces (APIs). These had to be added to each application before communications between applications could even be considered.

Adding an external API to an application (especially to those which had never been designed to have them) was an even more daunting task. It meant finding a place in the

application code that could provide access to data — while maintaining both the integrity and performance of the adulterated application.

Once this was done, today's familiar problems of synchronizing data formats, semantics, and events between programs remained. These were then individually solved through custom code, rather than via common integration services. On the one hand, we now have the advantage of being able to purchase packaged enterprise applications and so no longer have to maintain these complex programs. On the other hand, we have given up access to the special knowledge of internals that we used to possess — and so are entirely dependent on our vendors to provide the APIs that are 'appropriate' for our needs.

Yet, despite the advances, the fundamental problems that had inhibited application integration for years had not really been solved. For example, the business need for timely responses to business events ensured that batch data transfers between applications were usually unacceptable. When this was 'solved', the demand for ever greater systems interconnectivity (including between businesses and consumers via the Web) has only encouraged more changes to what businesses required.

New business imperatives

Today, in what was unthinkable a mere fifteen years ago, business managers seek to permit consumers and business partners to interact directly with their own business applications in an effort to become more responsive and, presumably, more profitable. Indeed, the drive towards real time business interactions, together with the nature of increasingly fleeting business opportunities, has created a need for IT to respond to rapid changes. No longer is this within an IT timeframe; it must be within the business timeframe and satisfy the strategic business requirement.

In a sense, therefore, we are on the threshold of a new world in which IT efforts move:

- **from well-defined computing operations**
- **to enabling transient business strategies.**

Given the critical importance of this, we must understand how best to achieve it. What we cannot afford is to repeat the software engineering mistakes of the past and, in this context, there are a number of problems that have stood in the way of application integration.

These remain barriers to IT's ability to respond (to the new requirements and timeframes of business) and include:

- **resource dependence**
- **precedence relationships**
- **complexity**
- **correctness.**

These problems arise because of the nature of procedural code. The impact is discussed below, and explains why:

- **the procedural approaches continue to frustrate; procedural code is inherently fragile**
- **pure declarative approaches, by avoiding these problems, are more robust and productive.**

Code fragility

Code fragility refers to those aspects of coding that are frequent causes of error and maintenance. In the 1950s and 1960s, several studies were done to determine the types of coding errors that were most commonly encountered. Among the most nemerous were:

- **errors of iteration control (such as coding or modifying loop entry and exit conditions correctly)**
- **case control (whether coded using if-then-else sequences or case statements)**
- **transfers of control**
- **consistent data typing in assignments**
- **levels of indirection (address versus content)**
- **data initialization**
- **sorting.**

These have changed little over five decades. Let us examine these from the perspectives of:

- **resource dependencies**
- **precedence relationships**
- **complexity**
- **correctness.**

Resource dependence

Procedural approaches explicitly manipulate computing resources, and most often in ways that are specific to the resource. Although efforts exist which help with the problem such as resource encapsulation (for example, printer drivers) and isolation layers (for example, data access services), these solutions require great discipline to implement.

Every programmatic dependence upon specific characteristics of a given resource introduces not only fragility but opportunities for new errors in translating the business requirements. There are several types of resource depen-

dence that need to be avoided, including physical data dependence, logical data dependence, platform dependence and connection dependence.

Physical data dependence

Physical data dependence occurs where you rely on a specific data organization, including its structure and location. Most procedural code is written to manipulate specific data organizations in specific ways, rather than the logical relationships among data. This tends to be rigid.

There are many reasons why physical data organization and its associated access methods may need to change. For example, the trade-off between storage efficiency and access efficiency is a well-known challenge. Selecting one over the other often mandates a change to the existing organization of data.

Similarly, there are many potential access methods for reading, modifying, destroying or creating data. Each of these has different performance and memory characteristics. While ordinary encapsulation goes a long way toward solving the problem of physical data independence, it does not provide selection of the best access method for any particular data structure nor address the more difficult problem of selecting the best data organization. Solving these requires automated selection via automated optimization techniques.

Logical data dependence

Logical data dependence occurs where you rely on how data is presented to users (and applications) and how they exploit it. For example, introducing a new use of data often requires changes to other programs. So long as no information is lost, including the logical relationships among data elements, there should be no barrier to changing data presentation for new users and new programs.

But this is not always possible. Solving it requires a standard data presentation model and the means to translate operations on this model into operations on any information-equivalent model (such as the physical, stored data).

Both are missing from procedural approaches. But both are native to most declarative approaches.

Platform dependence

Platform dependencies include both hardware and operating system ones. Few software applications (whether packaged or custom) are impervious to changes to the platform.

Device dependencies — such as changing a monitor, printer or a disk drive, let alone a network — often lead to:

- **serious application failures**
- **the requirement to make subsequent program modifications.**

This was generally true until the advent of standard drivers. While the situation has improved, it is not yet complete. Indeed it is the desire to minimize or eliminate this type of dependency that drives most of the interest in portability — leading first to language and interoperability standards and eventually to Sun's Java and Microsoft's CLI.

This procedural problem is far from being solved, however. What can be stated is that we understand most of the principles that should be followed (but we have yet to follow them).

Declarative approaches do not wholly solve this problem, either. They do, however, make it easier to address — by eliminating direct manipulation of physical resources.

Connection dependence

Connection dependencies include those on:

- **invocation methods**
- **inter-program communication methods**
- **inter-system communication methods**
- **device connectivity**
- **physical precedence relations (requiring that one piece of code execute before another, usually due to side effects).**

Over the years there have been tremendous debates (and many changed programs) over the relative merits of:

- **sub-routines**
- **in-line procedure calls**
- **remote procedure calls**
- **asynchronous messaging**
- **and so on.**

Similar debates have ensued over the best physical transport mechanism for inter-program and inter-process communication, including the protocols and physical resources. The problem remains, however, that connection dependencies abound in most applications.

Indeed, the bigger the application the more such dependencies tend to exist. Although careful layering of procedural code, and the introduction of standards, has helped,

the basic issue remains. Again, declarative approaches do a good job of encapsulating connection dependencies, permitting connections to be provided as a combination of simple nesting of declarations and transparent services.

Precedence relationships

Precedence relationships control the order of function invocation and completion. These may manifest themselves either directly through control code, or indirectly through data states.

Almost all coded order dependencies are implementation-specific. This means they can only be checked for errors in the context of the particular algorithms chosen by the programmer. Precedence relationships of this type are closely related to the twin problems of coupling and synchronization.

Another form of precedence relationship is inherent to business requirements, which have a strong process component. For example, orders usually precede shipments and deposits usually precede withdrawals. Changing precedence relationships of this type almost always requires changes to code when implemented procedurally (assuming they were correctly coded in the first place).

The problems of enforcing and changing precedence relations within code have not been solved. However, by expressing precedence relationships as rules (defined in a repository and implemented by either a rules engine or a process engine) the impact of changes can be greatly reduced. Because such declarative rules can be triggered by business events, rather than by conditions tested in code, they accommodate changes in business requirements and are easily maintained.

Complexity

As system size grows, the number of errors increases non-linearly. That is, when it comes to design and development, scalability fails. This fundamental consequence of software engineering is clearly related to the inherent limits in human abilities to conceptualize and remember large numbers of entities at one time and in one context. The solution to this problem is three-fold:

- **firstly, abstraction (or 'chunking') enables complex concepts to be treated as a single entity with understandable interfaces and behavior**
- **secondly, orderly decomposition permits complex problems to be broken down into manageable portions**
- **thirdly, function generalization can be used to reduce the number of such portions.**

Managing complexity lies behind various attempts to deliver efficient software engineering, including:

- **top-down, structured design and object oriented methodologies**
- **certain popular architectural approaches, like componentization and service-orientation (Web Services are merely the latest of these).**

Whereas control of procedural code complexity requires discipline and attention to detail, control of the complexity of declarative solutions is more automatic. Each declarative statement behaves as a unit of abstraction, similar to that achieved by componentization. With a uniform data model and strong typing, the 'contract' or interface of the declarative component is self-describing.

Correctness

Guaranteeing that code is correct is a difficult proposition. Traditionally we have sought to prove correctness through a combination of error checking within the code and software testing, both of which depend on completeness (no holes) or coverage.

Error checking tends to be limited to local state correctness, while software testing is more focused on functional correctness (producing a 'right' answer). Neither is very adept at catching the type of error which occurs due to either run-time environment changes or incorrect business requirement capture and translation during design.

In addition, software testing is particularly sensitive to component interaction complexity. When components can interact in almost any order, the ability to test all possible sequences of invocation becomes impossible even when the number of components is still quite small.

One partial solution is to require that components (especially distributed components) be stateless and that all shared data be transactional, so that component testing is path independent. Functional correctness can be enforced with integrity conditions or constraints, although:

- **few developers are inclined to code assertions or constraints**
- **most developers object to the (presumed) performance penalty imposed for such checks.**

Proving the correctness of a procedural solution, either

manually or automatically, is a very difficult problem. By contrast, declarative approaches are usually based on a mathematical model or algebra so that syntactically correct expressions are also valid semantic expressions. As long as each declarative statement expresses a correct assertion about the business problem being addressed, and the collection is complete, the solution is, then, guaranteed to be correct. The problem of debugging is significantly reduced. For example, most such errors that occur while developing RDBMS applications have to do with the procedural code that invokes SQL rather than in the SQL itself. Similarly, errors in developing rule-based systems usually are associated with failures to capture rules correctly (a requirements problem) or to declare all the rules.

Summarizing the procedural problem

In overview, the use of procedural languages creates three major problems, broadly speaking:

- **first, the more procedural the grammar rules, the more difficult the language is to learn — even if the language is graphical (consider modal versus non-modal graphical user interfaces)**
- **second, as discussed above, procedural elements tend to be the source of errors, causing high maintenance costs and functional rigidity**
- **third, because procedural elements expose physical organization and structure, changes to that physical organization and structure cause costly and error prone maintenance efforts.**

But there is also a fourth problem. This is particular to integration. Take all of what I have discussed above together, as is likely to occur when integrating applications, and integration using procedural languages cannot satisfy today's constantly evolving business agility requirements. The procedural approach is too rigid for business needs to be met. Something different is required — and this is not a general purpose language (GPL).

General purpose languages are insufficient

Most GPLs have a procedural dimension. Users must:

- **know something about physical implementations and resources**
- **specify step by step how to accomplish each task.**

If users need to access existing data, they need to know how that data is physically organized. In consequence, general purpose languages almost inevitably have a procedural element to them — meaning that they must be able to take advantage of order. (If this is not obvious, try to imagine the concept of the 'next' — or 'previous' — data element without those elements being ordered. Then try to imagine a completely non-physical ordering.)

Almost by definition an 'average' user will not know how to use the procedural elements of a computer language (Figure 5.1). This suggests that general purpose languages, like procedural ones, are also insufficient for modern business and application integration needs.

Figure 5.1: Procedural vs. Declarative

Procedural (simplified procedural function, in pseudo code) -- raising salaries by 15%

```
PROCEDURE NEW-SALARIES
FILEPOINTER *IFP
STRING IFORMAT, EMP-NAME, TITLE, BUFFER
FLOAT SALARY
INTEGER EMP-ID
IFP = OPEN('EMPLOYEE-SALARIES')
WHILE (NOT EOF)
{
      INIT-VARIABLES()
      RETURN-CODE = READ (IFP, IFORMAT, BUFFER)
      IF (RETURN-CODE == FILE-ERROR)
      BEGIN
            CALL FIXERR (RETURN-CODE)
            EXIT
      END
      IF (SALARY > 10000)
      AND (SALARY < 100000)
      AND (TITLE <> "PRESIDENT")
      THEN
      BEGIN
            SALARY == SALARY * 1.5
            RETURN-CODE == REWRITE(IFP, IFORMAT, BUFFER)
            IF (RETURN-CODE == FILE-ERROR)
            BEGIN
                  CALL FIXERR (RETURN-CODE)
                  EXIT
            END
      END
}
close (ifp)
exit
```

Declarative

```
UPDATE EMPLOYEE-SALARIES SET (SALARY = SALARY *1.15) WHERE
SALARY > 10000 AND SALARY <100000 AND TITLE LIKE 'PRESIDENT'
```

Moving towards the declarative

I would now like to return to a point I made earlier about rules-based integration components. These offer a clue as to how to avoid the problems of the procedural while delivering flexibility into the hands of those who need it most.

For example, dynamic rule-based routing — as found in many integration products today — provides the capability to direct a message to a particular recipient conditionally based on message sender, time, content and so on. In effect, rules are a kind of declarative integrity constraint on valid message routes. The result is that rules can be changed or augmented, without shutting systems down, as well as constitute a relatively simple language to learn and use.

Today, several implementations of rules provide a graphical interface for implementing and managing those rules (for example, in IBM's WebSphere MQIntegrator, SeeBeyond's

e*Gate, Sybase's e-Biz Integrator, etc.). With careful implementation:

- **messages can correspond to business events**
- **applications — that constitute both senders and receivers — can be identified with specific business functions.**

This raises the level of abstraction to something a business user might understand and be able to use.

Model process integration

But we can go further still. Model driven process integration moves beyond business rules, towards point to point message flows which enable control of entire processes. However, implementing a process which is a complex collection of point to point activities is difficult to conceptualize and can introduce process integrity problems.

By using a graphical process design tool to drive a process engine (essentially a sophisticated rule-based message router), the level of abstraction is increased. If both the design tool and the process engine support process abstraction and process independence, it becomes possible to implement business processes in terms of business activities and process flows that are recognizable to a business user.

Such a business process specification is tantamount to the declaration of a set of:

- **activity integrity rules that constrain the correct invocation as well as correct completion of business activities**
- **process integrity rules that define the precedence relationships among activities.**

The system then translates this declarative specification into an implementation with physical resources (albeit based on a manually defined mapping in most such products today). The advantages of this are that the problems of the procedural are avoided and the results are easier to understand.

The declarative vs. procedural

The creation of declarative languages (as an alternative to procedural languages) is driven in part by a desire to avoid (or at least to reduce the impact of) the procedural problems described above. Instead, to understand the power of declarative languages, suppose that users could concentrate on what they wanted to achieve rather than on how to obtain it. In such an environment, users would simply declare the goal. The initial state and the desired end would be defined by those constraints or conditions.

This approach requires that the system (not a user or programmer) automatically be able to:

- **translate the goal declaration into an optimized set of component procedures, that are invoked as needed**
- **produce a result guaranteed to achieve the declared goal.**

In contrast to procedural programs, declarative approaches (Figure 5.1) are:

- **easier to understand**
- **easier to write and modify**
- **significantly more succinct.**

Pure declarative languages have associated with them none of the problems described earlier. Indeed, they deliver the very benefits that procedural languages fail to achieve.

The declarative applied to EAI

In this context, the ideal declarative language for EAI will consist of two parts:

- **one for expressing integration goals and constraints**
- **one for specifying the characteristics of the physical resources that could be used, and object references (names).**

Such a declarative language would have nestable constructs and encapsulation enabling abstraction and controlling complexity. The language itself would be based on a closed algebra, so that all expressions in it are provably correct and unambiguous while the execution engine would have:

- **an optimizer, for best use of resources**
- **a scheduler, so that resource conflicts would be avoided (and load balancing achieved).**

The execution engine would access an active repository to translate user references into physical references. Data mapping and transformation would occur automatically as needed, whenever one business function was the precedent of another.

The view of the system, as seen by the user of a declarative language, is that it exploits terms and units which the user

understands. Declarative languages need, therefore, to be semantically rich so that users can accurately express goals.

Indeed, a declarative language is all about semantics or intended meaning. The software that processes a declarative user request must hide the platforms, physical data organization and the procedures that manipulate that organization. This means that fundamental operations must preserve information integrity — information is never augmented, altered, or lost except in ways that are specified explicitly by the user.

Integration requires many of the important features of a declarative language. Therein lies the declarative approach's power to deliver on the promise of EAI.

The procedural is not the language of business

All the above is not to suggest that procedural languages are useless. Nevertheless, we should limit their use because of the high price to be paid, especially in the EAI context where the types of facilities and operations required for an integration infrastructure are now reasonably well understood.

What is more important is that we now accept, and understand, that the business language which specifies integration is the language of business processes. Using a declarative approach is the developer's equivalent of the delegation of authority and responsibility used to implement most business processes. (Few managers want to repeat step-by-step procedural instructions to their employees.)

This mirroring of business management practices means that IT alignment with business can be more easily achieved. Although we may not yet have a pure declarative language for describing such business processes, we can today take advantage of those early declarative languages that are already available for integration. In addition to the declarative capabilities within these integration tools (such as are associated with rules engines and process engines, etc.), there are other opportunities:

- **nearly all data integration and data transformation effort can use an RDBMS to good effect, and with that RDBMS come the declarative capabilities of SQL**
- **simple rules engines and their tools can be used to avoid much procedural code (these tools generate or incorporate reliable procedural code based on a declarative specification).**

We are now facing a future with high training, platform, and maintenance costs that are aggravated by every use of procedural language. You should work hard to identify and reduce these costs by limiting the use of procedural language. For every element of your integration effort, question:

- **why a data transformation requirement should be met with procedural language**
- **whether, or not, you have resource independence**
- **whether falling back to a manual activity implementation (when the application server goes down) will break your process integration.**

Think of Web Services (especially when using them for integration). Recall what happened when you last encountered a broken Web link or a page that no longer has the information you thought you would obtain. This will be the fate of Web Service implementations: they are overtly procedural — the traditional programmers' haven of familiarity.

Now is the time to consider declarative languages and how much they can simplify business. Better yet, do not just dream. Instead, insist on introducing and using the declarative at every opportunity. Your business will benefit.

Management conclusion

The procedural approach is beloved of developers, largely because they have been taught this way and know little better. Developers often enjoy being clever and creative, and tweaking code for performance when it is unnecessary. Unfortunately, as Mr. McGoveran describes, it is the declarative approach that is best suited for a world where the ability to adapt fast matters. The declarative and the business world are natural partners.

For those wishing to see the emergence of 'business analysts' who can flex an organization's existing IT without need to be a part of that IT, the declarative is the way forwards. It has too many advantages to be, logically, resisted. Nevertheless, inertia and unfamiliarity will inhibit the adoption of the declarative. That will be a bonus for those who understand what can be achieved, but a loss for those unable to adapt or adopt the new.

Java is seven ..., and counting

Dr Keith Jones
IBM Worldwide Software Solutions

Management introduction

In spite of a difficult business climate this year, thousands of Java enthusiasts converged upon San Francisco in March for the seventh annual JavaOne conference. Millions of Java clients, from smart cards to cell phones to desktop computers and thousands of Java servers, are already installed worldwide and delivering value.

In the year since the last JavaOne, significant advances have been made across all the Java 2 platforms — not least with:

- ***XML capabilities, which have been added to almost all platforms***
- ***Java Web Services, which have been implemented by many early adopters.***

In this analysis, Keith Jones:

- ***reviews progress across a spectrum of major initiatives undertaken by the Java Community***
- ***addresses some popular criticisms.***

Platform of choice

At the JavaOne Conference this year in San Francisco a large audience — comprised of developers, vendors, consultants, analysts and the trade press — met for the seventh consecutive year, to discuss:

- **new advances in Java technology, from Java Card to J2ME/MIDP to J2SE to J2EE to the linkages to XML and Web Services**
- **its application to real world problems.**

In the seven years since Java was first introduced it has grown to be the most widely practised of all programming languages and fueled an enormous explosion of applications. These include real time ones, embedded ones, some desktop ones as well as those implemented on servers as large as mainframes.

In effect, Java has become the language of choice not only for developers of corporate presentation, business and data access logic but also for middleware product developers. Recent surveys have also shown that Java 2 Platform — Enterprise Edition (J2EE) is the platform of choice for the vast majority of corporate application server users when new applications are being considered.

That said, there are really several, albeit related, Java platforms. Each has its own growing base of standard specifications plus a broadening number of vendor implementations. All were on show.

This expanding universe of specifications, vendor products and mission critical applications is one measure of Java's success. Possibly the more important evidence, however, is that presented by the growing, and deepening, pool of skilled developers. A Java 'community' really does exist.

Platform development

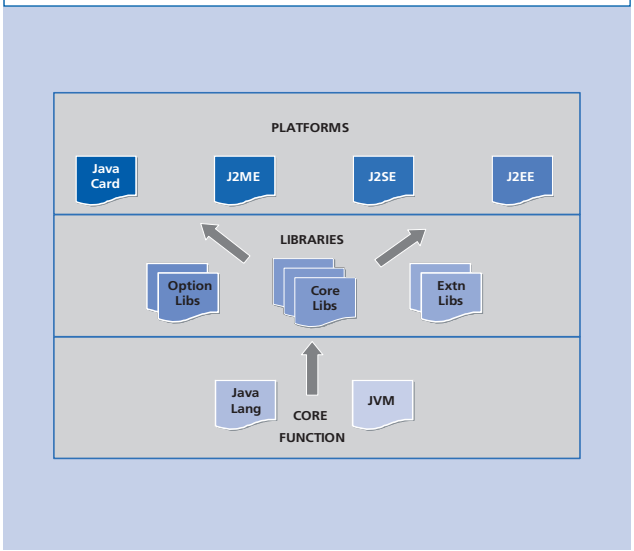
There has been significant development work on each of the major Java platforms in the last year. The various groups, involving participating company members of the Java Community, have agreed both new specifications as well as updates to existing specifications. In many cases these specifications are:

- **predated by proprietary and open source prototypes**
- **followed by reference implementations that are developed by open source organizations.**

New formalities have been agreed this year that give open source organizations access to early drafts of specifications, as these are developed. They will also obtain access to compliance test toolkits.

Figure 6.1 shows the relationship between the several types of specification being developed and maintained. Remarkable consistency exists across sets of these specifications. Although 'write once, run anywhere' does not necessarily apply to all Java application code — unsurprisingly enterprise business logic will not run on cell phones — Java skills do translate well across the major platforms.

Figure 6.1: Java specifications



Java Card

The Java Card 2.2 platform is close to reaching its final level of specification. An estimated 100 million smart cards have already been shipped that conform to some level of this open standard specification. A growing number of licensees — including American Express, Citicorp, Fujitsu, Gemplus, IBM, Mitsubishi, NTT, Schlumberger and Visa International — are deploying applications using Java Card technology.

Smart cards can securely store large amounts of personal information on an integrated microprocessor chip in many different forms, including one identical in size and feel to a credit card. Java applications can also be stored and, when needed, are loaded into the smart card's memory as bytecodes for interpretation.

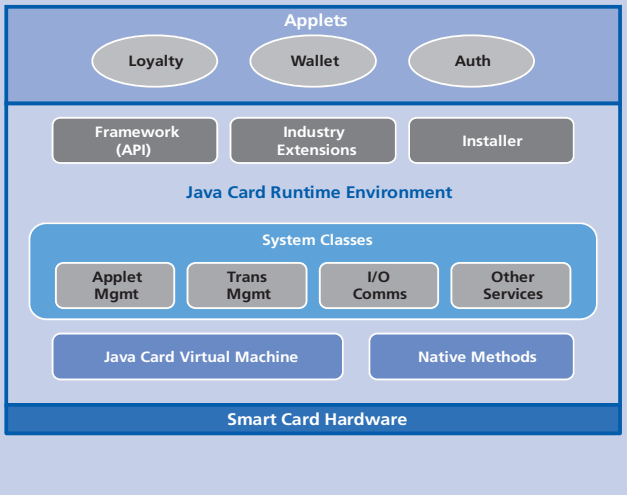
This combination provides a rich foundation for:

- **identity checking**
- **personal preferences**

- **medical data**
- **many other forms of 'presence functionality'.**

In addition, multiple isolated Java applications can reside on a single smart card (if required). Since the application logic runs on the smart card, there is no need to transport confidential personal data over unreliable or insecure network connections. Because the smart card execution environment is physically constrained it would not support Java applications designed for larger platforms. However, 'write once, run anywhere' does apply to Java card application code within the scope of available smart card hardware platforms that support Java Card technology (Figure 6.2).

Figure 6.2: Java Card architecture



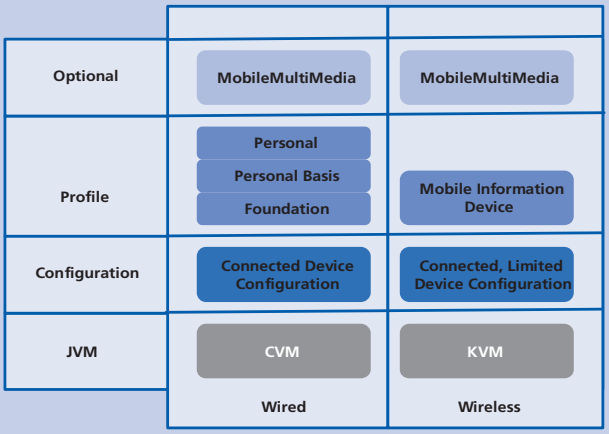
A subset of the Java libraries that exist on the larger platforms (J2SE for example) are available to application developers on the Java Card. This subset is tailored to the needs of smart card applications but is growing over time. Some library functions that are specific to Java Card Technology — such as on-card management APIs — are also included.

J2ME and MIDP

The Java 2 Platform — Micro Edition (J2ME) — has also made advances. An estimated 400 million J2ME capable devices have already been shipped; many are already running J2ME applications.

J2ME defines an architecture (Figure 6.3) for a wide range of consumer wired and wireless devices that share the constraint of relatively small footprint for both memory and processor cycles. While wireless J2ME devices are the most numerous at present, the market for wired devices — including embedded processors in cars, home and other appliances and set-top boxes for example — is beginning to grow.

Figure 6.3: J2ME architecture



Although these two families of J2ME devices share a common Java architecture, the Consumer Virtual Machine (CVM) for wired devices is more capable than the Kilobyte Virtual Machine (KVM) for wireless devices. The reason is simple: wired devices are often less constrained. Both in turn are smaller than a full JVM, as found on larger Java platforms. The basic configuration libraries — CDC and CLDC — also reflect this difference in platform capability and the connectivity requirements of the two families.

The Mobile Information Device Profile (MIDP) recently reached its 1.0.3 level. The next major level, 2.0, will be published soon. MIDP strikes a balance between size and functionality — in effect enabling providers to offer new interactive services on mobile devices. A version of MIDP is already available for Palm PDAs and several demonstrations of new 'midlets' on wireless devices were demonstrated.

Within families of devices, J2ME application code can be written once and run anywhere. With so many devices already shipped this becomes an extremely attractive proposition for service developers. Provisioning and management software was also on show from licensees, like Nokia and Motorola, keen to pave the way for this rapidly expanding marketplace.

In addition to games and information services (such as map

details and weather forecasts), new prototypes were shown that linked MIDP cell phones directly with Web Services hosted on J2EE application servers. For example, attendess saw transactional access to a (fictitious) theater ticket agency using a client subset of JAX-RPC APIs and a KSOAP engine enabled cell phone connection.

For wired devices several profiles of functionality have been developed:

- **the Foundation profile is common to all device types**
- **the Personal Basis and Personal profiles are libraries tailored to suit devices having different levels of capability (such as graphical resources).**

To set this in perspective:

- **some wired automotive devices have very limited resources**
- **in contrast, many digital TV set-top boxes are almost as capable as desktop computers.**

In addition to the prototype SOAP communications demonstrated on J2ME devices, broadening support for a subset of the JMS (messaging) APIs is emerging. This will enable users to connect directly to enterprise messaging middleware running on J2EE servers. In so doing they will obtain reliable message:

- **downloads from persistent stores**
- **uploads for store and forward transmission.**

J2SE

The Java 2 Platform, Standard Edition (J2SE) is the technology package with which Java developers are most familiar, not least because it is derived from the original 1996 JDK. The latest 1.4 level has already been downloaded by 1M+ users in the first month since its introduction in February 2002. J2SE has evolved to become a robust and mature development resource for universal Java clients.

This latest release contains an enhanced collection of functional libraries covering a wide variety of applications needs. The latest to be added are:

- **core APIs for logging**
- **XML document processing**
- **enhanced security**
- **fast I/O.**

With support for 64-bit memory access and 72-way processors, the focus for J2SE 1.4 development is on scalability and performance.

Most vendors and tool developers have already announced support for J2SE 1.4. The Java Community expert groups are now working on development of J2SE 1.5 — to be made available by the end of 2003. The focus for this next release includes improved:

- **monitoring and manageability**
- **scalability and performance**
- **integration support for XML and Web Services (these currently are available as 'add-on' packages).**

J2EE

J2EE Version 1.3 was first introduced in September 2001. It includes tight integration with XML standards, the J2EE Connector Architecture and Java Message Service (JMS) technologies as mandatory components. These features open up new opportunities. More relevantly, they leverage existing technology investments in a market already worth more than $2B in 2001.

J2EE Connector extensions already available include those for:

- **Customer Relationship Management Systems (CRM)**
- **Enterprise Resource Planning Systems (ERP)**
- **Enterprise Information Systems (EIS).**

There are already 13 J2EE 1.3 platform-compatible application server/tools vendors, including:

- **BEA Systems**
- **Borland**
- **Computer Associates**
- **Forte Tools**
- **IBM**
- **iPlanet**
- **Macromedia**
- **Pramati Technologies**
- **SAS Institute**
- **SilverStream**
- **Sybase**
- **Trifork**
- **TogetherSoft.**

All of the remaining J2EE licensees are expected to follow suit in the coming months.

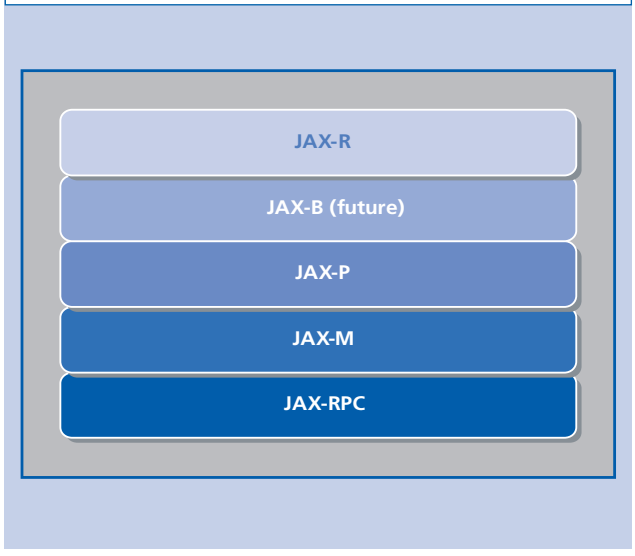
Many of the J2EE technology sessions at JavaOne were focused on maturing best practices, frameworks for programmer productivity and blueprints for using the rich functional content available in J2EE 1.3 vendor products. The J2EE application model is now firmly established as the deployment pattern of choice for most new applications. The addition of the J2EE Connector Architecture has greatly improved connectivity to legacy enterprise systems.

The major theme for advancement to J2EE 1.4 (in early 2003) is the integration of XML and Web Services. Java components — such as Servlets and JSPs, session EJBs and message-driven beans (MDBs) — will, in the future, be deployable as Web Service end points that are fully integrated in their respective Web or EJB containers.

Java and XML

Such is the importance of communicating XML documents, and then their processing on both Java clients and servers, that a special Java XML pack of library functions (with reference implementations) has been made available by the Java Community. The Java XML Developer Pack (see Figure 6.4) is currently being updated and released on a quarterly basis for use with J2SE and J2EE systems.

Figure 6.4: The Java XML pack



Nevertheless, criticism has been voiced about the fact that these functions are not yet fully integrated into J2SE and J2EE platforms. In counter-point it is fair to say that each of the APIs is based upon open standards that are still evolving — and will not be finalized for some time yet. In addition, although the standards for base XML documents are now relatively stable, there has been a shift from DTDs to XML Schemas for data type validation.

The XML Schema language is much richer — and more complex — than the DTD language. This has caused the groups responsible (within the Java Community) to take more time in defining the required Java APIs. When you consider what needs to be covered in this area, it is readily apparent why time is needed. For example, the Java API for:

- **XML Processing (JAX-P) is now at the 1.2 level — with full support for Schemas as well as for both SAX and DOM parsing and XSL-based transformations**
- **XML RPC (JAX-RPC) is the basis for all synchronous communications using XML to package data parameters flowing between systems and this API is the foundation for communications with Web Services that are defined using WSDL and based on SOAP 1.1 protocols; the JAX-RPC library also includes utility functions for automatically generating Java stubs and skeletons from Web Services definitions (for ease of use)**
- **XML Messaging (JAX-M) is a higher level API for sequences of essentially asynchronous one-way communications between systems that flow potentially complex XML documents (such as those used in B2B interactions); there is some optionality built into JAX-M in the form of profiles that support particular protocols — for example, communications with ebXML-compliant systems are supported as well as SOAP communications involving intermediate routing between end points**
- **XML Registries (JAX-R) is a special purpose library for programmatic access to diverse business registries; access to both UDDI and ebXML registries is currently supported (using consistent JAX-R APIs) and other registries will be added as open standards evolve and converge; this function also enables Web Services and peer to peer computing for the Java and J2EE platforms**
- **XML Binding (JAX-B) is not yet available but in late 2002 it will provide automatic mapping of XML document elements into Java components for easy processing; this API is taking longer to define than expected because of the inherent sophistication of the XML Schema language and the richness of the mappings required but will, when complete, remove much of the need**

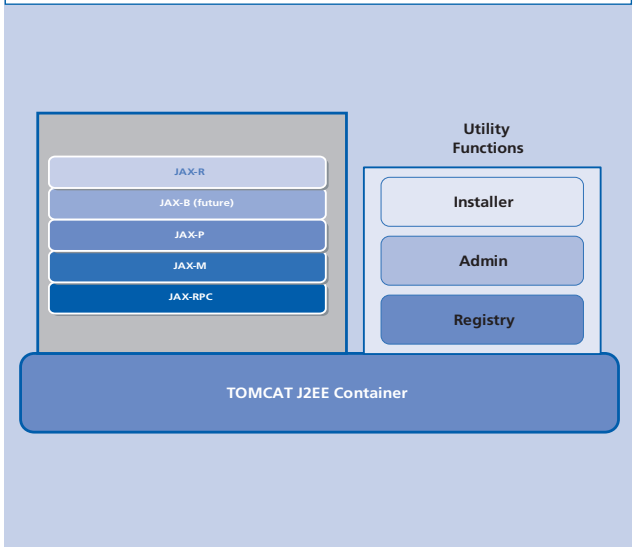
for navigation of document structure in Java Web Services business logic.

None of the Java APIs for XML assumes that communicating end points must be Java systems. This facilitates openness and access to Web Services that might be implemented on other platforms — such as Microsoft .NET systems.

Java and Web Services

The Java Web Services Developer Pack has been made available in 2002 for use with J2SE clients and J2EE servers to facilitate development of new Web Services. It includes the Java XML Developer Pack as well as utility functions and the open source Apache Tomcat J2EE Web Server and Container (Figure 6.5). Not shown are the Servlet, JSP and standard JSP Tag Libraries, which are also included in this pack for convenience.

Figure 6.5: Java Web Services Developer Pack



Using this pack, developers can create and test Servlets and JSPs designed to run as Web Services. A test UDDI Registry is included — to hold the Business and Service definitions — and the Java APIs for XML document handling are included to complete the function needed for real world services.

As applies to Java XML, this pack will be:

- **enhanced on a quarterly basis during 2002**
- **integrated into J2SE and J2EE platforms at their next revision level (during 2003).**

Success or omissions?

In my view, few could have predicted the success that the Java Community has had over the last seven years in:

- **defining a set of consistent platform specifications**
- **delineating an expanding marketplace for systems vendor products, middleware vendor products, professional services and skilled Java programmers.**

JavaOne illustrated that success. Projections are for continued growth and even acceleration. Yet there is always more work needed, and many argue that Java support for some critical systems functions is still missing or at best rudimentary. Security and management are undoubtedly such functions. While Java security has improved by giant steps in recent platform releases, yet more is needed to:

- **protect confidentiality of XML data**
- **ensure proper authentication and authorization for Web Service invocation (for example)**
- **define the management of large numbers of Java components in large-scale deployments.**

Critics also complain that 'write once, run anywhere' still does not (and never will) apply, as was promised when Java was introduced. However, Java code can be written once and run anywhere on platforms conforming to a particular architecture. My view is that it is not reasonable to expect that code written for one architecture should necessarily run on all others (even if these are 'Java'). Balancing these criticisms are demonstrable facts, which are hard to deny. The first is that the skills needed to write Java code on any of the major Java 2 platforms do translate well. The second is that the number of available skilled programmers continues to grow, and at a phenomenal rate.

Management conclusion

In one sense the 7th JavaOne was more of the same, with consistent and methodical progress. That is rather dull. But it is eminently satisfactory from a customer viewpoint.

As Dr. Jones has described, progress has been across the board — from Java Card to J2ME/MIDP to J2Se to J2EE to the linkages to XML and Web Services. While not all is perfect — for example, the XML capabilities are taking longer to pull into J2SE and J2EE than many wish for — the acceptance by people of Java as being their key skill is not diminishing; rather it is growing by leaps and bounds, which will make it easier to implement applications as new functions are agreed and delivered.

Free middleware: surprising facts

Tom Welsh
Consultant

Management Introduction

Writing middleware is difficult, time-consuming, and requires considerable expertise as well as resources for development and testing. It seems illogical to give the results away.

Yet there is much middleware available which is free. Furthermore, the volume and scope is increasing. In this analysis, Tom Welsh looks at what free middleware is available today, and why:

- ***Part 1, in this* MIDDLEWARESPECTRA, *sets the scene before focusing on free ORB middleware***
- ***Part 2, in a future* MIDDLEWARESPECTRA, *will focus on a range of other free middleware — including the likes of messaging, web and application servers, IDEs, databases and even development tools.***

Setting the scene

Chris Stone, co-founder and first CEO of the Object Management Group (OMG), often remarked that "middleware is software that no one wants to pay for". It does not take too much reflection to guess why. After all, the end user is hardly ever conscious of middleware — especially when it is doing its job properly. Why would anyone want to pay for something they cannot see and do not understand? "Out of sight, out of mind."

The people who appreciate the invaluable contribution that middleware makes to corporate applications are mostly software architects and developers. They do not usually buy, which explains why there is such strong demand for free middleware. But why is there a supply?

Software, including middleware, is usually made available 'free of charge' for one of three main reasons:

- **to create an installed base of users who are familiar with a product; eventually, some proportion of these may upgrade to a paid version — with additional features, performance or scalability**
- **as an incentive to increase the attractiveness of associated products which are paid for**
- **on principle, based on the belief that software should be shared and not sold at a profit.**

Implicit in the first of these is that every user who adopts the free version is a customer denied to competitors. Microsoft is the classic exemplar of the second — having piled dozens of free products into its Windows operating systems. Among these are middleware items like the Component Object Model (COM+) and Microsoft Message Queuing (MSMQ). It should not surprise that so many IT decision makers choose to use such products when they come 'free' with an operating system. This seems, in cost terms, preferable to paying extra for an object request broker (ORB) or a message oriented middleware package like IBM's MQSeries.

In the third, the open source movement has made great progress in five years. It now offers some world class software (much of which will be discussed in more detail in Part 2, in a future **MIDDLEWARESPECTRA**) including:

- **an operating system (GNU/Linux)**
- **an HTTP server (Apache)**
- **several RDBMSs (PostgreSQL, MySQL, etc.)**
- **various integrated development environments, from NetBeans (which is Sun-sponsored) to Eclipse (IBM-sponsored).**

The economics of middleware

Although free middleware always sounds attractive in these times of pressures on IT to contain or reduce expenditure, it is even more appealing. Against this is the reality that the capital cost of software licenses is only part of the total cost of ownership (TCO). Other factors that must be taken into account include:

- **the cost of paying development and operational staff**
- **training and the learning curve effect**
- **availability of staff**
- **quality (of code) and the ability to obtain fixes or upgrades**
- **maintenance.**

Last, but not least, middleware tends to support mission critical activities between applications and/or systems. What happens if a business critical application depends on a form of middleware which is discontinued or abandoned by its open source — or, indeed, proprietary — developers?

Not all of these factors are necessarily negative. It can be argued that the beneficial effects of the open source culture are more valuable than the purely financial windfall of obtaining something for nothing. In the open source world, if a package needs to be extended or modified, the source code is available. The user simply hires staff with sufficient expertise to do the work properly.

Similarly, if bugs cause trouble, there is nothing to prevent a user from tracking them down and fixing them. Indeed, if an organization feels it has a big enough stake in the success of an open source package to warrant the investment, there is nothing to stop it contributing development resources or funding to one or more such projects.

Free commercial middleware

Many commercial middleware vendors offer free (or very low-priced) entry-level editions of their products — usually in the hope that customers will upgrade to the more expensive 'enterprise' editions once they realize that more is needed to deliver serious projects. For example, Borland offers a free 'evaluation' edition of its VisiBroker ORB, and Iona does likewise with Orbix. These products do not fall within the scope of this analysis, as they are not really intended for production use.

Truly free commercial middleware packages begin with the ubiquitous, inexpensive infrastructure of the Web. While Microsoft's Internet Explorer and AOL's Netscape 6 are examples of free commercial Web browsers, it is harder to

point to a free commercial HTTP server. Microsoft's Internet Information Server (IIS) comes closest; it is bundled with the server editions of Windows NT/2000. Its intention is clearly to boost sales of Microsofts's server operating systems — by stuffing these with valuable features.

Most of Microsoft's other middleware, such as COM+ and MSMQ, is also bundled with Windows. With the advent of .NET, Web services too become a no-extra-cost Windows feature. To develop applications that exploit these, however, users must pay for Microsoft's Visual Studio.NET — at present the only IDE to offer comprehensive support for .NET.

On the Java side, Remote Method Invocation (RMI) and a slightly restricted version of the Common Object Request Broker Architecture (CORBA) are included in Sun's free Java Development Kit (JDK). By comparison to .NET, developers have somewhat more choice than is the case for .NET. As well as paid-for IDEs from vendors like BEA, Borland, IBM, Oracle, WebGain and others, they can opt to use open source IDEs such as Eclipse or NetBeans. However, as for .NET, the programming tools (Java, etc.) usually require a purchase.

Free open source middleware

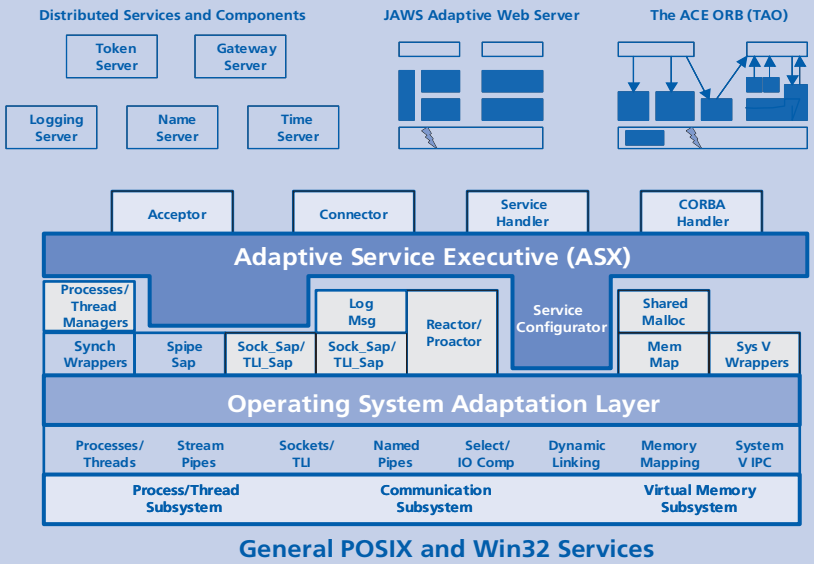
There is no easy way to compile a full list of open source packages. Moreover, it is difficult even to estimate how many copies of a given package are in use. Because these 'products' are freely available and cost nothing, most of the conventional tracking methods are frustrated. There is no revenue trail to follow, and casual downloads cannot be distinguished from those destined for serious use.

Several CORBA products are among the best-known free open source middleware. There are several reasons for this:

- **CORBA is fully standardized — and all the relevant specifications can freely be downloaded from the OMG's website without any formalities (not even registration is required); in one sense, CORBA can even be thought of as 'open specification software'**
- **CORBA is over a decade old which means that the specifications have had time to mature; moreover, the task of implementing the specifications to create working software is now much better understood**
- **unlike COM, or Java, CORBA can be used with over 20 languages and on almost any platform**
- **there are plenty of books, training courses, and information resources available around the world to make CORBA accessible.**

As CORBA expertise becomes more widespread and less of a black art, numerous free ORBs are springing up from commercial and academic sources. Some of these are faster, or better in other ways, than their commercial counterparts. Yet they may also have hidden constraints (less

Figure 7.1: ACE's relationships



maintainability, less strict adherence to standards, etc.). On the whole it remains true that you receive what you pay for.

That said, there are at least half a dozen open source CORBA implementations that bear comparison with the best commercial ORBs. Of the three products successfully tested and registered as complying with CORBA 2.1 under The Open Group's Open Brand for CORBA, two — MICO and OmniORB — are open source (the third is Fujitsu's commercial Interstage). To be fair, this happened two years ago. Several leading ORBs were not submitted for test because they complied with subsequent, incompatible versions of CORBA.

Free open source ORBs tend to be available for more platforms, partly because their definition of 'platform support' is less formal than that of commercial vendors. As well as posting compiled binaries for the most popular platforms, open source developers provide full source code. All and sundry are welcome to download the sources and compile them wherever they please. Any minor problems that arise can often be dealt with by a little judicious hacking — provided the person doing the hacking is sufficiently knowledgeable.

Thus the MICO documentation states that, in addition to running well on the main commercial platforms, "we heard rumors that MICO somehow works on the following platforms: OS/2 on Intel... DG/UX on Intel... LynxOS..." In a similar vein, TAO has been ported to a number of real time environments and MVS OpenEdition, over and above the standard platforms.

TAO and ACE

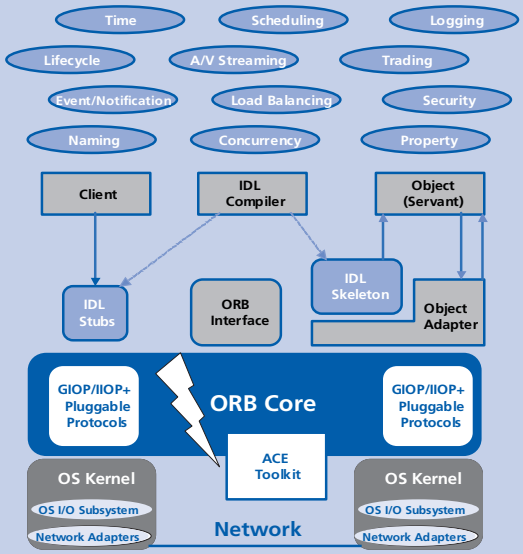
Short for 'The ACE ORB', TAO forms part of the Adaptive Communication Environment (ACE) . This is an open source distributed systems platform being developed at Washington University and the University of California at Irvine under the direction of Dr Douglas Schmidt, one of the world's leading middleware experts on the design of high performance middleware. The project has been funded by Boeing, Cisco, DARPA, Ericsson, Microsoft and other organizations — potentially testimony to its perceived value.

Thousands of developers have contributed to the code base, making ACE one of the world's largest open source projects, and probably the largest to focus on middleware. The latest release, TAO 1.2.1 beta, complies with the major features of CORBA 2.5.

Like other open source efforts, TAO is unconstrained by the needs to ship product, beat the competition's feature list or maximize quarterly revenue. This leaves the development team free to explore the potential of a given technical approach in an unhurried, methodical way.

Figure 7.1 illustrates key components in ACE and its relationships. Figure 7.2 shows the TAO architecture.

Figure 7.2: TAO architecture



Support is provided by Object Computing Inc. (OCI) of St Louis, Missouri and PrismTech of Burlington, Massachusetts and Gateshead, UK. OCI has been supporting TAO since 1999, while PrismTech — which also markets a product line of CORBAservices and J2EE products — announced its entry in December 2001. Both companies also support JacORB for those customers who prefer a Java-based product (to date, TAO supports C++ only).

PrismTech's Total CORBA Solution (TCS) is the more ambitious initiative, comprising JacORB and TAO together with PrismTech's own set of five CORBAservices (OpenFusion), packaging, support and consulting. TCS is by no means free — although PrismTech claims that it can reduce ORB costs by at least 40%. But the ORBs themselves cost nothing. What customers pay for are the CORBAservices, product packaging and support (which is available 24x7 if necessary). So confident is PrismTech in its business model that it has set itself the extremely challenging goal of becoming the global CORBA market leader by 2003.

Although TAO itself is available free of charge, and users have full access to its source code, it is complex. But OCI has found no shortage of demand for its services. Business development manager Malcolm Spence freely admits that "[w]e tell them up front, we're competing with your ability to fix this problem". If an organization that adopts TAO finds it more convenient and effective to set up its own internal support team, that is fine. On the other hand, some prefer to buy in the necessary expertise, as well as training, documentation and mentoring.

The United States Department of Defense has been showing an interest in CORBA for some years, and several government-sponsored reports have cautiously recommended its use (interoperability and branding were the main caveats). Today CORBA is central to a number of mission critical projects.

For example, the nine month Open Systems Avionics Technology II (OSAT II) project culminated in a convincing demonstration of real time flight control software. An AV-8B aircraft, with a mission computer system comprising two PowerPC processors connected by the TAO ORB, conducted six successful bombing runs. On one of these a simulated fault took one of the processors offline in the middle of a bombing dive. The other processor took over so quickly that the Head Up Display (HUD) did not miss a single frame — meaning that the ORB recognized the loss of processor #2 and shifted all processing to #1 within 20 milliseconds. Boeing intends to apply the software to planned updates of the mission computers of the F-15 and F/A-18 under the Bold Stroke advanced avionics initiative.

MICO

In the tradition of recursive acronyms established by GNU ('GNU's Not UNIX'), MICO stands for 'MICO is CORBA'. The intention of this project, based at the University of Frankfurt, is to provide a freely available and complete CORBA 2.2 implementation for educational purposes. 'Bells and whistles' have been deliberately avoided, together with proprietary techniques and dependencies. MICO aims to provide everything that is needed for full CORBA compliance, but no more. Its features include:

- **IDL to C++ mapping**
- **Dynamic Invocation Interface (DII) and Dynamic Skeleton Interface (DSI)**
- **an interface repository (with a Java-based graphical browser)**
- **IIOP support**
- **dynamic Any**
- **interceptors**
- **Portable Object Adapter (POA) support.**

MICO also provides implementations of the Interoperable Naming, Trading, Event, Relationship and Time CORBAservices. For users preferring a Java ORB, its authors recommend JacORB.

The team has extended an invitation to any and all to contribute to MICO — for instance by writing implementations of further CORBAservices. Alcatel is one major company that has accepted this invitation, funding the development of a CORBA Component Model (CCM) implementation based on MICO.

Full source code for MICO is available under the GNU General Public License (GPL). Binaries for AIX, Digital UNIX, HP-UX, Solaris and Linux are sold by dpunkt.verlag für digitale Technologie of Heidelberg and Morgan Kaufmann.

JacORB

JacORB is a free CORBA 2.3 ORB written in Java by Gerald Brose, a research assistant at the Freie Universität Berlin's Institut für Informatik. It was designed to be simple, portable and comprehensible, and has been used for teaching purposes. It is, however, also fast and efficient. Benchmarks show JacORB throughput similar to RMI for small payloads — and significantly faster for large ones.

JacORB is being actively developed and provides a strong set of features including:

- **an IDL compiler**
- **IIOP**

- **an interface repository**
- **Implementation Repository**
- **BOA**
- **both DII and DSI**
- **request-level and message-level Interceptors (also known as filters and transformers).**

There are also implementations of the Collection, Concurrency, Event, Naming, Trading and Transaction CORBAservices. A Security CORBAservice implementation is planned.

JacORB is written in 100% Java, and offers both CORBA IDL and Java-only distributed programming. Multithreaded clients and servers are supported through various threading models. The ORB comes with full source code, examples and documentation under the GNU Library General Public License (GPL), which grants the rights to copy, distribute or modify its code.

JacORB has been well received, with documented applications by Motorola, Philips and Siemens among others. It shows signs of becoming the open source Java ORB of choice: both the TAO and MICO developers recommend it to those who prefer not to code in C++.

omniORB

omniORB is a high-performance implementation of CORBA2. It was developed at the Olivetti and Oracle Research Laboratory (ORL) in Cambridge, England. It is now owned and developed by AT&T Laboratories, at the same location.

In the developers' words: "We are extremely keen to promote the spread and use of CORBA. To this end, we have made omniORB2 freely available in source form and have so contributed a significant component of software engineering technology to the community. We believe the simplicity and efficiency of our implementation is complementary to commercial offerings, and that it will appeal particularly to the research and academic communities. We encourage you to use it, port it, fix bugs in it, extend and generally improve it."

Despite the mention of academic communities, omniORB is a robust product that has demonstrated excellent performance in at least one independent benchmark. A somewhat dated report from the Distributed Systems Research Group at Charles University, Prague, showed omniORB to be faster than Chorus CoolORB, Iona's Orbix, and two other commercial ORBs.

This is no accident. omniORB has been carefully optimized with a view to using it in time-critical systems. It includes a C++ language binding and support for IIOP and has been tested for interoperability with Orbix, OrbixWeb, and VisiBroker for C++. The runtime is fully multithreaded in a platform independent manner. A Naming CORBAservice implementation is supplied.

omniORB has been freely available in source form since May 1997 under the terms of the GNU GPL and LGPL. Solaris, Linux, Windows NT and Digital Unix are supported. Third parties have ported omniORB to several other brands of UNIX, OpenVMS, NeXTStep and at least one embedded platform.

Management conclusion

As well as the various free commercial middleware offerings, Mr. Welsh sees a variety of free open source packages that compare quite favorably with their commercial competitors. In the CORBA market, one of the most mature middleware sectors, there are at least half a dozen reliable open source implementations — notably JacORB, MICO, TAO and omniORB.

That said, open source middleware is definitely not for everyone. More sophisticated users, however, can benefit from the reduced up-front costs and superior flexibility of free open source packages so long as they realize that up-front licence cost considerations should not dominate acquisition decisions. The total cost of ownership is more significant — which explains why the benefits of free open source middleware are somewhat less than many might anticipate (or desire).

Nevertheless, in the ORB world, there is a genuine reduction in overall cost — particularly capital investment — and without any obvious loss of features, quality or performance. Quality support is also available from a handful of companies specializing in the field (for example, OCI and PrismTech).

The less quantifiable benefits of open source software should not be forgotten, either. User organizations gain more control over the functionality of their applications and the fixing of bugs — provided they invest in hiring suitably experienced developers to do the work. If they are willing to contribute to the development effort, users can often exert greater influence on the future direction of open source packages than they can with commercial products.

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 1460-7220