C3B Consulting's

INSIGHT-SPECTRA

Applying common sense to using technology intelligently

www.insight-spectra.com

Volume 24 Report 4; July 2011

Data volumes continue to breed like rabbits

Charles Brett, C3B Consulting Ltd.

Management introduction

One of the horrors of IT is how data continues to breed like rabbits. It is one of the issues that most concerns IT, not only because of the physical implications but also because of the cost implications.

This is not a new issue. But it is one that continues to accelerate out of control. Yet managing it is akin to herding an army of ants.

In this brief refresher, Charles Brett looks at the pressures and requirements and what this means for enterprises. Permitting data to breed out of control is no longer a cost efficient option for organizations. Yet hiding data under the carpet (to continue to mix horrible metaphors) is all too often the reality.

To start

Let me start with a simple proposition. Your enterprise has 1TB of 'normal' data – say traditional, transactional business data, forgetting about all the other information on PCs, etc. — that you need to run the business (most medium and larger business will have way more than this, but using 1TB makes for simpler math).

But 1TB is not 1TB. It is not even 2TB once back up and — depending on disaster recovery (DR) policies and other options (for example, snapshots, operational disk efficiencies) are taken into account — it will not even be 4TB. It may be much, much more. How much more ('the Data Storage Factor') will only become apparent if you dig deep. Remember, this is not about the technologies to store your data (SSD, conventional disk, tape, optical, etc.) but about the volume of data saved itself.

Considerations and challenges

When thinking about data, there are a plethora of issues (and the ones listed are only a selection) that should come to mind:

- backup
- operations
- DR
- compliance
- operational efficiency.

Each of these possesses its own characteristics, many of which interweave. For example, what is the backup policy? Is it to have one backup (which Apple seems to favor via iTunes for 'iDevices') only? Or is it to have clear generations and tiers of backup (as happens in most enterprises) — yesterday, last week, last month, etc?

Then there are operational considerations. If primary systems go down (those that affect the capability of the business to operate), then the imperative is to recover and restart as quickly as possible. This may involve, for example, regular snapshots so that the minimum that has to be done can be done. Then there is also the reliability consideration: how often do 'backups' which are thought to be reliable prove unreliable, incomplete or useless? This is what has driven the need for more backups, on the principle that more are better than fewer, for safety's sake.

Now add DR. IT Disaster Recovery has been adopted by most organizations. At its simplest, replica data is stored 'somewhere else' so that if a disaster occurs — it does not matter what — then not all is lost. DR at a minimum doubles the data volume that needs storing. But it may be more, depending on the policies adopted and an enterprise's specific requirements.

In today's world we also cannot forget compliance. Legislators and regulators want more and more to be kept in a form that can be readily accessed and not modified (to the eternal joy of lawyers armed with legally enabled discovery). This adds a further (usually unquantifiable) aspect to the increase in the amount of data that has to be located somewhere and safe.

Finally, at least for this analysis, there is operational efficiency. We all know that, on our personal PCs, we should not run them with less than 20% 'unused space': this is to ensure reasonable performance. In large enterprises, with high throughput data environments, it is not uncommon to see disks that are delib-

erately run at no more than 30%-50% 'data occupancy', because this is what assures the performance that users and applications want.

What does this mean?

Let us make some simple policy assumptions:

- backup is done daily over a 7 day week, and 30 days are kept
- 3 generations of the total past month are kept as further backup
- a full copy of the existing and previous 3 months is kept off site (DR).

If we start with our 1TB, that means 7TB by the end of the week and 30TB by the end of the month. Add 3 generations of past months, at its simplest, and this is 120TB. Add the DR and you have 240TB in total. What matters here is the DSF, in this case 240 times the original 1TB.

You may argue that a DSF of 240 is unreasonable. For example, if only 15% of data changes each day, you would reduce the data that changed to .15TB/day which amounts 'only' to 2.0TB at the end of week 1, and after 30 days only some 6TB. Now add the 3 generations of monthly backup and you have 24TB; add again the DR and it is 'only' 48TB.

Yes, a DSF of 48 is much better than the crude 240. Yet it is horrifying enough (and what would you think if you were buying and were told that you would need 48 times as much again in order to be safe?).

However, we have not included disk efficiency (though this will likely only apply to the primary disks). Nor have we included snapshots (which may add 10-20% per day depending on the operational efficiency requirements). Nor have we addressed what compliance requires: is this additional (for safety) or is it assumed to be covered by the backup strategy described above (there are plenty of compliance executives or administrators who will argue that operational data and its backup must be kept separate from compliance copies).

Finally we have made no allowance for growth nor for the multiple new types of data arriving. Twitter messages may only be 140 characters each but there are an awful lots of tweets that may be relevant to your enterprise's social network image. Similarly voice and other media are becoming ever more relevant.

The critical issue is to understand what is the DSF applicable to your enterprise? Only when you know what this reality is can you hope to improve and to manage your data volumes while still remaining in business and compliant with legislative and regulatory requirements.

What to do?

If data storage factors like 240 or even 48 seem excessive, they probably are — they are used to make the point. The good news is that there many techniques and technologies that can assist — but these are relevant only if you understand the size of the complete data challenge before you start to implement 'solutions'.

For example there are various flavors of device and/or software on the market that do automated de-duplication, which can radically reduce what is required. In addition. Then there are devices which can compress/decompress data on the fly: this again reduces the overall space needed.

A different approach is to rethink backup. The purpose of backup is data safety. If all data is demonstrably safe — using techniques like RAID — then backup is not the issue that it was (though DR is just as important).

For example, IBM's XIV product is a combination of low cost hardware components combined with sophisticated modularization and software designed to provide high reliability, high speed access to data storage but without needing to back up in the traditional way. Yes, copies are needed for DR but if XIV is deemed to be reliable, then the 3 past generations of backup can be removed (and similarly if the DR has an equivalently dependable store).

Using virtualization techniques is also applicable. There is more and more interest in taking virtualization beyond that for servers and desktops and applying its capabilities and management approaches to data. NetApp provides an obvious example, and there are plenty of alternatives.

Storage in clouds is a different, possible way forward. This can work if the cloud provider has done its homework and can deliver the facilities and capabilities to lower the DSF (it is in its interest to do so because making optimum use of storage potentially improves its bottom line).

Management conclusion

Data is still breeding like rabbits, just as it has always done. Data types that need to be stored dependably are exploding. What is more, different generations of data (and tiers) are also breeding, through IT having to act to ensure enterprises can continue to operate as well as satisfy legislative and regulatory requirements. Enterprises are faced with an explosion of data that runs and runs, if only traditional approaches are adopted.

While enterprises (and this applies to large, medium and small) do not need to go as far an introducing Myxomatosis (as happened in Australia to try to control its excess of rabbits), enterprises do need to start understanding:

- *what is their 'data storage factor'*
- *what rethinking can then be undertaken to contain data without compromising the business.*

Charles Brett
C3B Consulting
www.c3bconsulting.com

Cloud computing: an update

Amy Wohl, Wohl Associates

Management introduction

It is 2011 and cloud computing has been around now for 'several years' — which avoids trying to offer an exact date for when it started, because all dates are controversial. However, for practical purposes, many will consider the start as being when Amazon opened its EC2 service in 2006 as a public beta, or in 2008 when EC2 became a full production service. The underlying concepts of cloud computing, of course, are much older — dating back to the 1960s.

In this analysis, Amy Wohl examines what has been happening, and what to expect.

Much has occurred

In recent times cloud computing has moved from a new (and somewhat questionable) idea to a mainstream computing offering. That has happened because, even as cloud computing was maturing (I rate it as a teenager now) and solving some of its problems, other important things were happening:

- more companies were entering the market with additional styles of cloud computing
- the global economy went through a serious recession and 'making do' with flat or diminished budgets made the lower cost of cloud computing very attractive.

Now, let me talk about what customers can buy, and what they are buying. The early opportunity in the Cloud Computing market was the use of very well-priced computer power and storage from organizations such as Amazon with that EC2 service. Many companies, small and large, tried cloud computing:

- first by doing development
- then testing on these services.

It is possible to do most any computing task (with a few exceptions) on such services. But when buyers were going to do a great deal of work or if they required a more controlled or customized environment, most decided to move on.

At that time, the cloud computing buyer then had two basic choices:

- buy space on a public cloud that offers the services his or her organization was looking for
- set up a private cloud.

What happened?

Thus far a large number of enterprise cloud computing users have been going down the mainly private cloud route. This offers them complete control and

any degree of customization they desire (and are willing to pay for).

For many enterprise buyers, private clouds look like extensions to their data centers, but with the attraction that these extensions should require less capital outlay and/or less technical resources to implement and run. Of course, major private cloud providers, like IBM, will do it all for a buyer — identify the right applications, build the cloud and manage it. Nevertheless, many buyers want to control the cloud themselves once all is up and running.

With cloud services, public clouds and private clouds all winning customers in the marketplace, it has now became highly important to cloud computing buyers to be confident that they will be able to integrate their various computing environments. The term that has become popular for these multiplatform environments is 'hybrid cloud computing'.

This has its ironies. In reality there are no 'hybrid' clouds, but rather tools and services that enable data centers, public clouds and private clouds to be integrated. How exactly this occurs depends on each buyer's needs.

For example, an organization with a very large database (with many updates) will not be served well by placing the database in the cloud — it will take too long to upload and then update the data. In this sort of instance it is better to keep the data in the data center and run the application in the cloud — the hybrid cloud environment. On the other hand, a company that wants to communicate with its customers and create a community for them around its products would be well served by a private cloud. It can have confidence in its confidentiality.

Market sizing

The global cloud computing market is already large (estimates run in the neighborhood of $25 billion for 2011 and $114B in 2016 from some analyst projections — for more data go to: http://blog.goaruna.com/tag/cloud-market-size/), although the bulk of the market in both time periods remains in SaaS sales. One difficulty is that it is frequently hard to see the beyond this (clouds are rarely transparent).

As you might guess, there are differences in projections by analysts, with IDC forecasting rather less ($72.9B in 2015) than most. But all seem to be in broadly the same ball park.

That said, the real news may well be found outside the SaaS market which is noisy and global about its products. Infrastructure offerings are harder to describe: try explain the difference between a customer who hosts his infrastructure with a cloud provider and a customer who purchases virtual private clouds from that same provider. To complicate matters, one must also calculate the related management services that will be provided to both of these 'options' as well as estimate what customer usage growth rates will be, year after year.

As if this was not enough, try to project the growth of regional cloud providers — smaller vendors and service offerors who only exist in a specific parts of the world like Western Europe, China, India, Japan and South America. These are generally unknown outside their specific region. You cannot count what you cannot see.

In fact, in the view of Wohl Associates, they may be on the low side. It is much easier to talk to all the SaaS vendors and ask them for projections than it is to identify and then to talk with all the infrastructure vendors, many of whom fly under the radar.

In addition, there is ample software for the SMB market, available on-line, with little or no need to have an internal IT resource for access. Software for the enterprise market is also booming, with many products, both familiar and new, coming to the SaaS platform.

All the data indicates something about what customers are buying in the cloud computing market. This may be summarized as:

- lots of SaaS
- though hidden, much IaaS and PaaS.

Marking shifts

The extent of SaaS purchases indicates a shift from software installed on-premises to the customer's own hardware to the use of cloud-based applications. Some of this, of course, is new software — in essence applications that could not exist except as cloud-based software. An example is a software service which tracks customer activity on a community site, in enormous detail, so clients can use that informa-

tion to plan future campaigns and target offers to the right customer.

IaaS and PaaS are not so popular. But, over time, these are moving from the heavy interest in private cloud infrastructure to more assured use of public and virtual private clouds.

As for applications, customers are using cloud computing for nearly anything you might imagine. CRM and ERP are particularly successful.

In the future Wohl Associates see much collaboration, with every major vendor now offering an on-line collaboration tool. Anything that has the need for distributed usage and conversion (such as financial roll-ups for subsidiaries scattered across the globe) will attract. We also see lots of business process software being offered as SaaS.

Management conclusion

It is always a good idea to understand why many customers plan to have at least some of their applications move into a cloud. The benefits they perceive include:

- *the need for less skilled staff to design, implement and manage in-house applications (scarce skilled staff can be channeled to company-specific software, where real competitive advantage can be achieved)*
- *shorter time to value:SaaS software (and PaaS and IaaS infrastructure, for that matter) can be implemented much more quickly — often in hours or days instead of many months — which means the results can be used more quickly (marketing campaigns started sooner, new products offered sooner, revenues arriving earlier, etc.)*
- *the ability to collaborate with colleagues worldwide and/or to organize customers into communities*
- *lower costs because there is no need to acquire hardware and software or to implement new systems and manage them on an ongoing basis; design, implementation and ongoing management (such as installing upgrades) are built into the cost of a cloud computing environment.*

Wohl Associates does not see cloud computing as a temporary fad. Rather it is leading to a profound permanent change in computing that will alter the face of how organizations provide and support infrastructure and applications. It will level the playing field between small, medium and enterprise-sized companies.

Remember also: in the cloud, everyone has access to high quality, professionally supported applications. This is the one dimension that you forget at your peril.

Amy Wohl
Wohl Associates
www.wohl.com

Thinking systematically about offshore service delivery

Peter Bye, Bye Associates

Management introduction

A feature of the past decade or so has been the rapid increase in the numbers of IT-skilled people in South and East Asia, as well as in Eastern Europe and Brazil. India is probably the most notable and widely discussed example. In all cases, the unit costs (e.g. man-days) of the labor are lower (or much lower) than in areas such as Western Europe, the United States and Canada. Therefore, one option often considered for cost reduction in service delivery, including software development and support, is to use an offshore organization for at least some of the work.

In this analysis, Peter Bye considers how well using offshore resources works. He references offshore organizations that are delivering part or all of a service for an 'onshore' client. He develops a simple model to use as a framework for assessing the cost/benefits of using an offshore approach.

Where you cannot offshore

There are some services which cannot use an offshore component because of the nature of the customer's business; basing a decision on cost alone does not enter into consideration. Examples include defense, policing and other sensitive government agencies like tax and security services. In most cases all this work has to be done onshore and usually on site. Security clearances are also frequently necessary to work in these environments. Clearances normally require residence in the country for an extended period or even nationality of the country concerned.

However, that still leaves many possibilities where offshoring is possible. My concern is with the nature and management of the relationships between the client and the provider, in particular for development and support services of basic software (such as operating systems and compilers), application packages and custom applications.

A simple service structure model

Most often a service is modeled as a client and a provider (Figure 1):

- the client is represented by IT and other project professionals working in the same location as the ultimate customer that is requesting the service and the client is responsible for analyzing the requirements, and specifying the service and its deliverables; the specification is then passed to the provider
- the provider then develops the specified product, where 'product' is used to mean the deliverable: in the context of this analysis, the product could be in any of the three categories introduced above, or support services associated with them (for instance fixing problems or adding new features).

During the provider's development phase, the client must manage the project, monitoring progress, solving problems and handling changes. This requires ongoing interactions between the client's project management and the provider's management.

This interaction is shown by the double-headed arrows between the client and provider in Figure 1. Such interactions may also require a number of iterations around specification and development, represented in the figure by the curved arrows.

Following delivery by the provider, support may be required — both on the client side and by the provider. How much effort is required for support and for any deployment depends on the nature of the 'product'.

There is a wide spectrum of possible ways of delivering the service according to this model:

- at one extreme, the client and provider could be in the same locally-based team, or even be a single person
- at the other extreme, the provider could be an offshore organization within the same company, or even a separate company contracted by the client.

The total cost of service delivery is the sum of the

costs of each of the components shown in Figure 1, namely analysis and specification, product development, project management, deployment and support (5a client and 5b provider).

Evaluating the costs

To justify product development offshore, or even from a distant location onshore for that matter, the total cost must be less than the cost when local product development is used — that is where one team, comprising client and provider, are co-located. Care is needed in the analysis of costs because remote product development may considerably increase the cost of other components. Obviously, the cost reduction in development must not be eaten up by the cost increase attributable to such other activities.

Specification is particularly important — because communication between the client and a remote provider is much more difficult than local communication. Ambiguity or incompleteness in specifications too often leads to serious problems during development or deployment.

A report on a research project on offshore development highlights this point (Note 1). This project used 15 carefully selected, highly qualified project managers, with experience of 135 offshore projects between them. On the question of specification, one of them remarked that 'you will get exactly what you asked for, so you better (sic) make sure you are asking for exactly the right thing'.

Developing basic software such as operating systems and compilers is probably the easiest to do remotely, as the requirements are generally consistent across different types of user. In fact, basic software development centers are often remote from most of their customers. Compilers in particular have precise language specifications, and the target machine architecture is equally well specified. As long as the provider has expertise — an important caveat, as I will discuss later — in compiler or other basic software development, problems should not be too severe.

Similar considerations apply to application packages, as any development center is often remote from most of its customers. Additional factors add complications not present in basic software. For example, how much awareness of the target market and domain-specific environments is necessary? Even an understanding of simple factors such as branch sort code structures for banking packages can cause problems. I am aware of one case where just this point caused a major delivery problem: the specification assumed an understanding of the UK branch sort code structure; the offshore developers did not understand the implications or practices.

Custom applications need special care, as it is almost always near-impossible to specify an application precisely enough: local development enables extensive communication between client and provider to compensate for ambiguity or missing detail; whereas remote or offshore development reduces the scope for extended discussion; much more formality is needed.

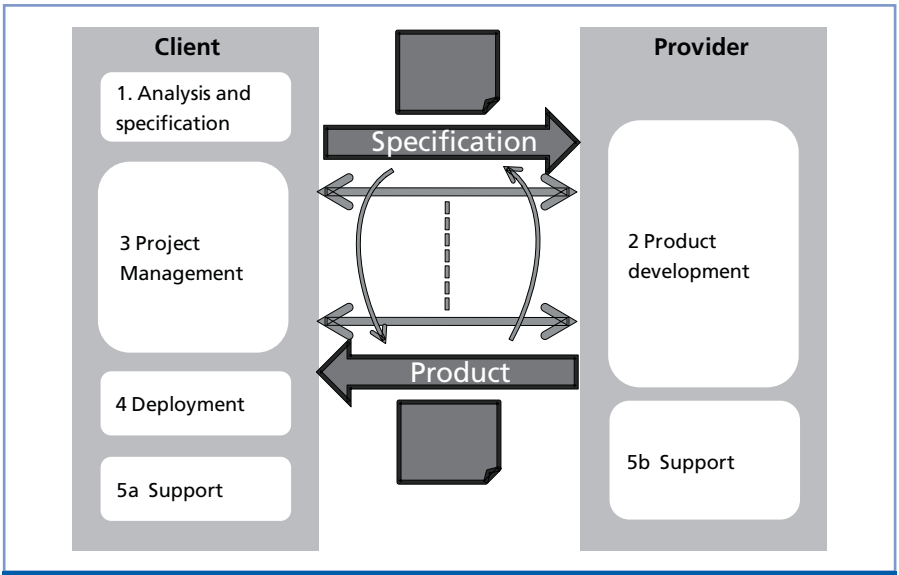

Figure 1: The service model, as 'client' and 'provider'

Working with a provider in another country compounds problems, as language and cultural barriers may inhibit communication. That well-known 'barrier of a common language' between English and American English is only the most obvious example. (It should be noted in passing that Agile software develop-

ment requires that developers and those requiring the product are co-located. Even here, cultural and language problems may be encountered if groups are from different countries.)

Inadequate specification can also lead to support problems later on; the resulting product may not be fit for purpose and therefore require extensive fixing — a process that may consume significant extra effort (activities 5 a and 5b in Figure 1). To make matters worse, there may be no budget allocated for such fixing — particularly for 5a — which then means cost overruns and unhappiness. The problem of unexpected support costs is not hypothetical. It occurs in real projects.

Avoiding pitfalls

In the above discussion of costs, I do not intend to imply that consideration of offshore work should be avoided. Cost analysis has to be performed systematically, to see what possibilities (and probabilities) exist. It should never be assumed — as sometimes appears to be the case — that lower development unit costs (i.e. costs per man-hour) lead to a reduced overall cost. To put it another way, to assume that the only aspect that changes when going offshore is that costs will go down is naïve.

A number of inter-related factors are at work here, all ultimately affecting cost, which can lead to serious difficulties if not openly addressed. There is a considerable churn in employment in many offshore organizations, an understandable situation as people strive to improve their lot: churn affects costs by bidding up salaries as people hop from one job to another and it leads to extra training requirements, when people who have been trained move on to better things, taking their newly-acquired skills with them.

Which brings me to a major point concerning skills: my experience suggests that the time required to replace an onshore development and support organization with an offshore one is frequently significantly underestimated. Product development organizations have (in many cases) been established for an extended period, and employ highly-experienced and skilled people. Furthermore, there may well be close contacts with customers, leading to a deep understanding of requirements and how to satisfy them. These skills are not quickly acquired so a rushed move offshore can lead to a serious reduction in the quality of output and hence to customer dissatisfaction. Such problems are compounded by the employment churn mentioned earlier. The value of any apparent cost reduction is, therefore, likely to be more than offset by the overall loss of quality and performance.

Management conclusion

The apparent availability of large pools of skilled, low-cost resources is a magnet to organizations hoping to reduce costs. Relocating some or all functions offshore has to be taken seriously. However, there are significant potential problems, which Mr. Bye believes are often overlooked.

Custom applications are particularly tricky, because the physical separation of client and developer requires extra-careful specification of the requirements. The real costs expected during and after product delivery must, therefore, be realistically assessed at the start of any project. (in practice, these should be the same considerations that would apply with a remote relationship onshore — it is the separation rather than the onshore/offshore distinction that matters.)

For basic software and application packages, the key point is to recognize that a transition to an offshore group takes time and effort. The long-acquired skills in some development centers — including deep relationships with clients — are not easily replaced.

One possible approach, argues Mr. Bye, is to maintain overall architectural and technical guidance in the onshore team while gradually moving other functions offshore. Another is to be systematic in the approach to evaluating the possibilities of offshoring.

Finally, problems must be recognized and faced. Wishful senior-management thinking about cost reductions often seems to replace the necessary careful, disciplined consideration and planning that need to be made.

Peter Bye
Bye Associates

Note 1: "A risk profile of offshore-outsourced development projects" C L Iacovou and R Nakatsu, CACM Vol. 51, Number 6, June 2008. There are many other studies as the subject does arouse great interest.

Value-based case study #10

GEVA: re-engineering an already proven payments system onto JEE using a model driven software factory

Management introduction

GEVA Business Solutions (based in Aachen, Germany) is a software company which specializes in the development, integration and distribution of payment transaction solutions. It is primarily, but not exclusively, oriented to German, Swiss, Austrian and Luxembourg financial services companies for the exchange of payments between each other. Its 50+ customers include Commerzbank, BNP Paribas, DEXIA and the German Federal Ministry of Finance.

Most payments systems have been built in either COBOL or using something like the .NET Framework. GEVA's original solution was no different. In the late-2000s GEVA understood that it needed to reset its solutions if it was both to offer constantly improving capabilities to its customers and at the same time reduce the cost of servicing those customers with their many different interfaces and infrastructures while still delivering an utterly dependable and proven payments capability.

In this value-based case study Uwe Klatt, the General Manager (Geschäftsführer) of GEVA, discusses:

- *the decision processes*
- *the selection of a model-driven, methods-based development partner (Integranova) that generated code for JEE*
- *the experience*
- *lessons learned, plus best practices.*

The business problem

GEVA was founded in 1982 as an offshoot from a local University in Aachen. Initially it focused on mostly hardware solutions for the finance sector before it refocused in the 1990s on delivering software. In 2004 a management buy-out concentrated the company on the provision of payment software applications which GEVA has successfully been selling to financial institutions to support electronic payment (e-payment) transactions between banks.

Our solutions are not, however, for end user customers. They are for the interchange of e-payment instructions and transactions between banks. As such our solutions have to be wholly robust and reliable: our customer banks depend on the throughput, accuracy and availability of our software.

In the late 2000s we started to find banks beginning to ask for Java, especially J2EE (now JEE), based solutions for payments. Payment systems are, as you can imagine, a sensitive area for banks: if these do not work as expected the results can be costly, for all.

Therefore, when the banks were asking about Java, it was more than just about asking for the introduction of a new technology. No purchases by banks would take place unless and until JEE solutions could be shown to come from an established, dependable vendor — one that possessed the credibility the banks would need in order to buy. This 'concern' is one reason why most banks are so slow to change any of the core aspects of their systems.

Technology requirements and implications

At the time that this interest in Java-type solutions emerged, most payment solutions were written in either IBM COBOL or using a framework like Microsoft's .NET. Furthermore, each bank had, and still has, its own distinctive combination of interfaces and infrastructure.

In the past this meant GEVA had, essentially, had to support at least one 'programming technology' (COBOL or C++/.NET) and sometimes both. Often these would be used in different combinations with different interfaces and communicating with different infrastructures within each customer bank, never mind between banks. As you may imagine, this was expensive and complex — which was not what customer banks wanted long term.

We listened to what our banking customers were saying. We thought about what they were suggesting and decided we could do what they asked. But to achieve this we decided that we wanted something like a model-driven software factory where we could

introduce a common application base framework that would generates JEE-based applications which could cover 90% of our existing core e-payment functionality.

In addition we wanted to improve our capability to deliver customizable solutions, ones where functions would be changeable or alterable very fast with a minimum of development (and a minimum of re-proving that what had changed was functionally and operationally reliable). If we could do this we would introduce a modern, JEE-based e-payment core product while reducing GEVA's exposure to complexity to only the minimum that was unavoidable — the interconnection to each customer bank's specific interfaces and IT infrastructure.

We also decided that we would:

- use our own domain expertise in payments and e-payments, basing this on our existing functional capabilities
- not to try to develop a new framework-based JEE core on our own: we concluded that we needed a partner that had both the Java/JEE experience but, as importantly, offered a model driven software factory development capability to deliver a descriptor framework that would be functionally provable, re-usable and facilitate change.

Selecting a partner

Finding a partner capable of meeting our requirements — this was late 2007 and early 2008 — was not straightforward. In Central Europe at that time there were few possible partners even supporting JEE at all, never mind understanding the financial services domain: most continued to prefer what the financial institutions already used — COBOL, .NET, C++, etc (this 'deafness' to what the banks were asking for we saw as an opportunity for GEVA).

We did, however, finally identify three possible partners with a potentially suitable, model driven, software factory approach. We performed the usual due diligence and held a 'beauty contest'. We decided that a Spanish company — Integranova, based in Denia and with a close research affiliation with the Universidad Politécnica de Valencia — best suited our needs.

In part this was because its MES (Model Execution System — Figure 1) was already oriented to JEE. In part it was because MES brought to us the structured factory approach we desired, where we could:

- rapidly capture detailed payment process descriptions in the MES descriptor tool (and keep this within the model)
- use those process descriptions to drive the automated generation of JEE code which

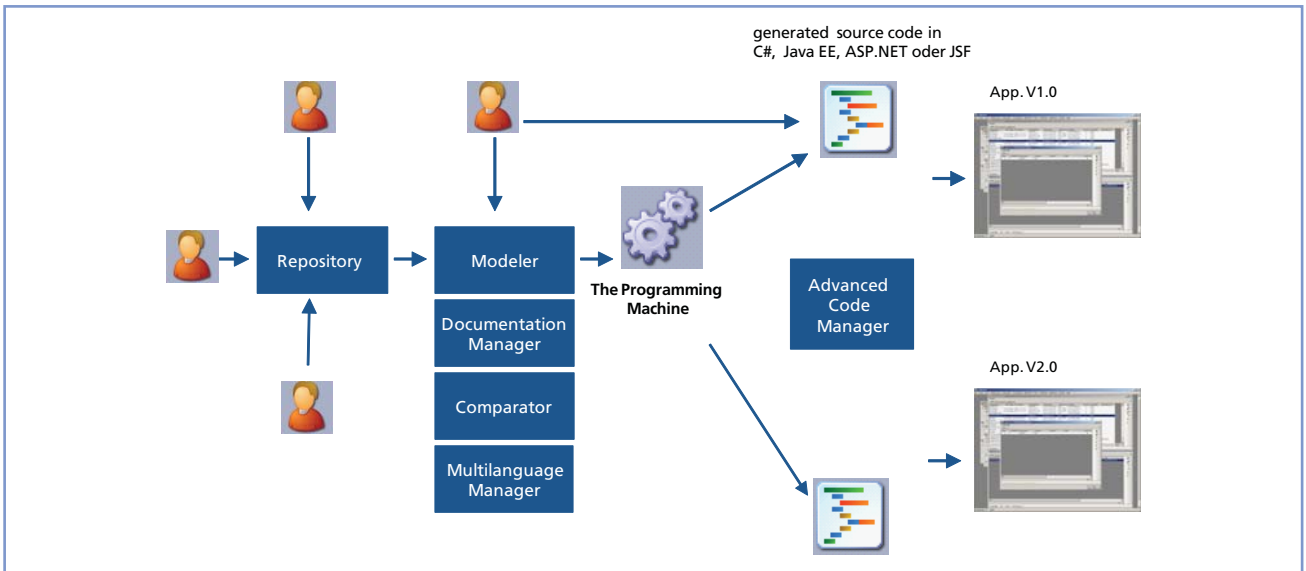

Figure 1: MES — a model driven software factory

would then become the basis of the new GEVA core e-payments platform.

But as much was to do with our decision to choose Integranova was its management willingness to be a partner and to work with us to re-engineer our existing, as well as new, e-payment capabilities into the MES. This willingness was clear from the start — and it has continued.

Progress and customer deployment

In early 2008 we started our partnership with Integranova. We now have about 80% of our original e-payment functions (originally written in C++/.NET) working in JEE via the MES. We hope to have the remaining 10% not yet done (remember 10% is bank-specific and has to be individually written) finished by the end of this year or early 2012.

We already have three banks using our new JEE-based payments platform, even though it is not wholly complete. These banks are using the JEE applications in production and this is providing the necessary confirmation of the needed stability and reliability.

We have many more banks waiting for us to complete, so that they can make similar transitions. Indeed we expect to move most, if not all, of our customers to the new platform in the next 2-3 years.

Going to JEE is, without doubt, giving us the competitive edge we hoped. We are now much further forward in satisfying the banks' wish to have Java-based solutions than the competition. We have also found that the 'agile' development approach works well in the new environment that we have introduced with the MES.

In addition, one of our previous business constraints has been removed. Before working with the MES our e-payments platform was pretty monolithic. Using the MES and working with Integranova we have re-architected and re-engineered our core applications/platform. In so doing we introduce a new flexibility. We have the possibility to 'break out' specific functions and to sell these to new customers, and even to new markets, as separate from the whole platform: this we could not do before.

Let me give you an example: we can, if we want, now separate the SEPA payment function and offer this to corporate customers (a new market for us) so that these corporates can make SEPA payments more easily. This has the potential to be a substantial commercial step forward for us.

Lessons learned

Moving from C++/.NET was not trivial. While GEVA had the e-payments domain knowledge, it did not have JEE experience when we started. This is why Integranova's tools plus Java and process modeling knowledge have been so important.

We have also learned that making JEE perform to the levels our customers need is not trivial with JEE (some customers are processing 1m-1.5M e-payments per banking day). Such volumes stretch JEE. For those relatively small number of our functions that demand such performance we have had to do coding by hand rather than use the MES model generated JEE code (the process descriptions are still in the MES). We are making good progress, however, and we now believe we understand the issues. But what we have learned — both GEVA and Integranova — is that it is not always simple to resolve complex, high performance transaction handling and performance issues in JEE.

On the less positive side, we have always aimed for a single JEE framework. Today, however, we have to have three different implementations of our platform — because of differences in vendor implementations of JEE on the main application server platforms. These different 'implementation flavors' are for:

- the IBM WebSphere application server
- the jBoss application server
- the Oracle application server.

We do not like this. We will have to address it in future development steps. We can hope that Java developments will help us. But we are going to try to identify the key differences between the three JEE application servers and then have Integranova reflect these in the MES so that each specific flavor can be nominated when we go to generate code.

An area where we do want the MES to improve (and Integranova is working on this) concerns how one moves from the modeling process descriptions to the generated JEE code. This works. We must, nevertheless, be able more easily to go from the process flow definitions described in the model to building up the

classes and the interfaces that our customers require. This would significantly improve our effectiveness and productivity for the future.

A fundamentally positive point has been the impact on developers. Compared to the learning curve that developers had for COBOL or C++/.Net before they became productive, it is now much faster and easier for new developers to contribute. The MES has significantly assisted and reduced the time needed for developers to start to contribute. This definitely improves our ability to integrate new people into our business.

Best practices

The most important best practice I can immediately think of may seem obvious. It is that understanding what a customer really wants, especially in terms of process flows and interfaces, is vital. If the understanding is not deep enough, mistakes can happen which are then costly to revisit and correct. Putting sufficient effort into this is essential if you are not to waste the effect that something like MES offers. Looking back we probably did not put as much effort to start with into this as we should have. Now we know we must always do this.

Related to this is that we have found we must spend more time with each customer to understand their flows and interfaces. When we do this properly, the development of that last 10% (the integration to each customer) is both faster and better. It is this last 10% which takes the greatest time (and cost), now that most of our core functionality has been implemented in JEE. This last 10% is the most difficult part to control. We think the best practices that the MES model-driven basis enforces will also help us here eventually (though not as much as is already happening with in our core functionality).

Another best practice that comes with the MES approach is understanding how the emphasis moves from development to focus more on business analysis (the description of the customer's payment process flows) and on architecture (which combines the e-payments core functions with the needed interfaces at each customer bank). The degree of involvement by developers has changed, helped by the automation of code generation. The best practice is to make sure your people change (and do not retain 'old habits') and that you encourage process analysts and architects.

My last best practice is also, in some ways, obvious. Choose your partner with care. While not everything has been perfect with Integranova (what relationship is ever 'perfect'?), Integranova has pretty much done all we wanted. It has grown good at being close to us and at assigning good people to be in Aachen with us. This makes a big difference to how you work together. Distance too often leads to misunderstanding or incomplete understanding — which both partners have managed (mostly) to avoid. The result is proving productive for GEVA over what is now a 3+ year-old relationship.

Management conclusion

GEVA selected Integranova because it offered a model driven software factory approach that generated JEE code. This made sense, not least because GEVA did not have the Java/JEE experience/knowledge necessary to re-engineer its e-payments platform on its own.

By 2012 GEVA will have a JEE-based, dependable core e-payments platform that is readily customizable for individual customers. In converting to the MES, GEVA obtains the ability to reduce (though not necessarily to eliminate) traditional coding.

For GEVA, the model driven factory approach enables it to work at higher levels internally (and also with its customers) to capture its own and its customers' payments processes and interfaces. Though the full commercial benefits have yet to arrive for GEVA, they are imminent as GEVA introduces its re-engineered JEE-based e-payments platforms to its existing customer base and starts to open new business opportunities.

INSIGHT-SPECTRA

is published and distributed worldwide by:

C3B Consulting Ltd.
19 St. Michael's Road
Winchester SO23 9JE
UK

Telephone:
+44 787 233 4000
+34 686 116 993

Email:
insight-spectra@
insight-spectra.com
or
charles.brett
@c3bconsulting.com

World Wide Web:
www.insight-spectra.com