C3B Consulting's

INSIGHT-SPECTRA

Applying common sense to using technology intelligently

www.insight-spectra.com

Volume 23 Report 1; January, 2010

Will AEP ride to the rescue of BI as well as confirm the reducing importance of the RDBMS?

Charles Brett, C3B Consulting

Management introduction

One way (if a little provocative) to think about the differences between BI (Business Intelligence) and AEP (Analytical Event Processing) has an analogy with aircraft 'accidents':

- *BI is akin to crash investigating – looking at the causal evidence and circumstances after the crash event and then trying to explain what happened; this is valuable and necessary but hardly desirable*
- *AEP, in contrast, is akin to ongoing engine and airframe systems monitoring, seeking to measure and anticipate and thereby preventing the incidents that aircraft crash investigators would have to research.*

Staring a discussion

In the edition of Pund-IT (q.v. www.pundit.com) of November 4th a dialogue started discussing "Traditional RDBMSs Stagnate as Analytical Event Processing (AEP) Emerges from the Shadows". This provoked a range of comments (q.v. Merv Adrian's blog (www.mervadrian.wordpress.com) about various aspects of the futures of AEP, RDBMS (Relational DataBase Management System), and business intelligence (BI) and RDBMS, all of which have inspired further consideration about BI and AEP – in addition to the original comments on SQL-based RDBMSs.

Strangely, since one of the opening comments was "that the relational database is a much more endangered species than I had thought. That is not to say that RDBMSs will disappear, or that their revenues will diminish." Several of those commenting on Merv's blog seemed to think that this was suggesting the opposite – that the demise of the RDBMS as we know was imminent. Of course, the RDBMS (and its associated revenues) will not evaporate — but the pointers are that the central relevance of the RDBMS will likely decline *relative* to newer initiatives, one being AEP.

Other readers suggested that it was not helpful to create a new label or acronym, such as AEP (which I had not created, having heard it from others). Yet, while many can agree that excess labeling is about as useful as percentage inflation — can 125% of effort truly be worth more than 100% effort? — analysis supports the notion that AEP does indeed reflect a set of different characteristics that assists understanding and therefore merits the differentiation.

How are BI and AEP different?

BI is not AEP. AEP is not BI. RDBMS is neither. Although they may one day complement or even fully reflect each other, today they are different.

Why, or how, are they different? BI is — at least traditionally — about the analysis of post-processed data - data that has been through the transaction or other preparatory mill (including ETL — Extract, Transform, Load). For the most part, it is harbored or parked in data warehouses, in a structured format ready for sophisticated analyses, most often delivered through some form of SQL-based tool.

Over the years an enormous amount on money has been poured into data warehouses by large corporations seeking an analytical edge, which to some degree BI has provided. But the reality is that much of the data that makes its way into data warehouses is out of date or, at best, stale by the time it has been through the process mill. This does not mean that practical conclusions cannot be drawn. They can be and are, but not in a timely manner.

In effect, BI has survived and prospered primarily because there were few alternatives. A poor analytical solution was better than nothing, and was bought for this reason.

To understand why AEP is different, think about what business and government customers want from analytics in, say, a credit operation. Not just a Visa or MasterCard — this also includes telephone credit cards or even mobile phone operations. Credit providers want to ensure that the credit being requested by a customer has an acceptable or permissible probablility that it will be repaid (and so the credit can be granted). Credit providers wish to avoid

issuing credit where the user has no credit, as well as seek behavior patterns indicating potential fraud. The end objectives are to issue 'credit' only where it is warranted while also withholding it when not appropriate.

Such real-time processing (in-flight processing is possibly a better description) across one or many data streams is not what traditional BI offers. However, it is what AEP does. Ergo, traditional BI looks exposed when compared to something like AEP — in its many forms, whether on financial markets, in industrial production, in credit handling, in flight engineering operations, in ship and port scheduling (there is a short and simple example from the BBC describing the latter — see Note 1 for the URL).

What makes AEP different? It enables analysis to be done as and when it matters by those who are most concerned. In many instances this is long before data 'hits' any formal RDBMS or arrives for ETL-like activities prior to being 'put away' (like the linen) in BI's holding warehouses.

AEP is less structured than BI, yet it also offers analysis before any transactional implication becomes necessary. Whether it is analyzing financial events in the search for arbitrage possibilities before making trading decisions (thereby starting the transaction sequence all the way through to the data warehouse) or examining sensor-originated data before issuing an alert for remedial action, the analysis of events takes place in advance of traditional processing. Fundamentally this is what separates AEP from BI (and arguably from both RDBMSs and even Business Activity Monitoring (or BAM)).

Analysis

Though many may not like to consider or admit it, BI has stagnated over the past 20 years. Once fashionable (as AEP will no doubt prove to be at some point), BI became costly, complex and involved much IT expertise – not least, the magic arts of ETL. This, along with its inherent time delays, has distanced BI from common business users.

Instead, BI has become the preserve of specialists seeking to wring from past data precious conclusions that are relevant for now (and the future). The closest analogy is with forensic scientists (in this case 'information scientists') delving for insights. Or, put another way, BI analysts might be characterized as office-based examiners of what has happened, who hope to shape the future.

In contrast, AEP (business) analysts are already much closer to the business. Look at Starview or Sybase's Real-time Analytics Platform to understand AEP's how's and why's. AEP is about 'new' (pre conventionally processed) data rather than 'old' data (post-processed and only now ready for analysis). It is significant that most of those looking to AEP are already front-line business people who are operating on or near the shop, pipeline or trading floor, who also possess practical business engineering insights and requirements.

If you accept this differentiation, all can move forward. That they are different does not mean that they cannot coexist. They can. But to achieve this, both approaches must be reconciled in terms of timing, storage and manipulation:

- pre-processed data — AEP's events — need not be 'thrown' away; events represent data points that can be mined for additional insights (they are not just on-the-fly happenings that have no longer term value)
- post-processed data (BI's traditional fare) is too often used behind the times; it can be spiced up by combining new and old data to bring revelations and possibly return the immediacy that BI seems to have lost.

At the moment, however, there is little in common in the two approaches which means a significant division remains. In the analysis of C3B Consulting, BI will need to adapt to what the AEP and CEP and streaming vendors (from IBM to Oracle, from Streambase to Sybase, from Progress to TIBCO — among others) offer rather than the other way round. This position has been confirmed in consultations with BI vendors about acquisitions, because these vendors understand what could happen to BI — it would lose out).

Management conclusion

Do not confuse the fact that several of the AEP vendors are also RDBMS and BI vendors; what they sell for AEP is different from that sold for RDBMS/TP or BI. As such AEP may yet rescue BI — if BI adapts.

No, the RDBMS will not disappear, though it may lose relative importance. Yes, BI is in danger, but it can recover if vendors and specialists wake up. No, AEP

will not either solve every problem or replace all other solutions. Yes, the overall market 'pot' will grow. But that growth will come from where in-flight data analysis assists the business. Expect AEP/CEP to drive a major part of that growth.

Notes

(1) Ships can now be tracked with smart software. The global shipping trade generates much data. Ships will likely move in and out of European ports much quicker thanks to that smart software which now monitors their daily movements: http://news.bbc.co.uk/2/hi/technology/8413566.stm

Charles Brett
C3B Consulting
www.c3bconsulting.com

In Brief: what did Cloud Computing achieve in 2009 and what will it achieve in 2010?

Robert Cohen, Cohen Communications Group

In 2009

In 2009 Cloud Computing became more than a service that small businesses or start ups would use. Big enterprise users showed that they were moving to put well-used applications into the cloud, often relying upon external cloud service providers. Early in the year, several large telcos announced cloud service offerings, including AT&T, Deutsche Telekom and British Telecom. By December, a flood of global carriers had announced cloud service offers — including Colt, Telecom Italia, Telefonica, SK Telecom and Chunghwa.

A few of the early service contracts that relied on clouds included GlaxoSmithKline's (GSK) choice to move about 100,000 employees from Lotus Notes to Microsoft's hosted platform for email, corporate communications and live meetings. Genentech, the California-based biotech company, put 16,300 of its employees' emails on Google's platform to obtain email, word processing, spreadsheet and calendar applications. Eli Lilly noted that it was running critical analytic applications in Amazon's clouds, setting up 'tailored computing environments' to run serious applications for research.

Some analysts were surprised to find in 2009 that many big corporations were making plans to subscribe to infrastructure as a service (IaaS) and put many of their applications outside traditional corporate IT centers. Indeed it seems that a quarter of corporations are already spending or planning to spend on IaaS, usually contracted with a services provider.

Besides 'big pharma', many finance houses started to build in-house cloud facilities as well as spending on external clouds. This shows how far corporations came in their thinking about clouds in 2009.

In 2010?

In the coming year, I expect this trend to continue and accelerate. I see several new ways to extend the current patterns:

- first, a number of the pharmaceutical companies and banks will complete their internal clouds; this will certainly show others that the shift from non-virtualized, inefficient compute centers makes sense
- second, telcos will begin to create more attractive offers for businesses; this may involve some innovative links between cloud pioneers and telcos which offer very high level service agreements that guarantee almost continually accessible service (for example, I would not be surprised to see a Salesforce.com link to a Verizon or AT&T)
- third, telcos will create ever more sophisticated global compute and storage networks to handle significant demands for cloud services from global corporations
- fourth, we may also see an 'iPhonization' of cloud services, where some cloud providers develop links to app. developers who can rapidly create cloud versions of business or consumer applications

- fifth, some vendors (think IBM and others) are making bold plays for enabling pre-packaged, in-house (or private) clouds; these offerings will only improve, reduce in cost and make the 'old' ways of offering enterprise computing look tired, inflexible and expensive.

If even only 2 or 3 of the above happen, 2010 will see dramatic advances in the use of clouds. In our analysis this will be most striking at the enterprise level. Look to see what new uses are adopted as the year rolls on.

Robert B. Cohen
Cohen Communications Group

More thoughts on 'BigDB'

Nick Denning, Diegesis

Management introduction

In the November 2009 **INSIGHT-SPECTRA***, Peter Gassner of Veeva Systems discussed what he would like to see from a 'large database' — what he called 'BigDB'. This made various people sit up. In particular he identified the need for a database capable of handling huge amounts of storage which possessed the following properties:*

- *a robust SQL interface*
- *transaction control*
- *good utilities*
- *high performance — without the need for tuning (no indexes, no parameter configuration and no special utilities to do what is obvious)*
- *scalability.*

Diegesis has been working with the Kognitio's WX2 database product in the context of specialist data mining database products to manage complex queries. In this analysis, Nick Denning offers additional thoughts, including that WX2 technology may be Peter's 'BigDB'. He brings to the analysis a background in risk management and middleware as well as deep use of databases.

What is Kognitio WX2?

Boiled down to the essentials, Kognitio WX2 is an SQL Server database without indexes. In a standard relational database one organizes tables with a primary index which controls the order of data on disk. This provides fast access by primary key to records, with secondary indexes providing alternative rapid access to data. The result is that it is usually the case that when the query plan reports that a table scan is to be undertaken, you expect a performance impact and then spend much time trying to 'design out' these plans.

Kognitio WX2 only does tables scans. A conventional table scan generally means massive I/O as the table is read from disk into memory on every pass whenever the table is too big to fit permanently into memory. This is usually the case because most familiar products were designed many years ago for single machines with limited memory.

Kognitio's WX2, however, was designed as a massively parallel memory resident database which typically runs on a series of blades each with a number of multi-core CPUs and fully populated memory plus a local disk (or mirrored pair of disks, typically about 4 times the size of the memory. The blades are connected by a Gigabit (or 10 Gigabit) dedicated network backbone. Consequently, when a query is passed to WX2, the query plan first analyses the query into its separate nodes.

For each table referenced in the query, in parallel, WX2 then determines which columns to project and the basis on which it will restrict its rows by a table scan. The data may be memory resident or may need to be first copied from disk. The projected columns of the table are copied into memory buffers spread

across all blades and table scans performed on each table segment. In this way, all the bottom layer nodes of the query are executed in parallel and hence with a minimum of elapsed time. The next layer of joins is then executed to produce new data sets again in parallel until each layer is returned.

Clearly such a mechanism is not appropriate for queries for single rows (as is often the case in transactional systems). Thus it is probable that we will continue to use traditional transactional database technologies (DB2, ORACLE, SQL Server, etc.).

However the more complex are the queries, the greater the performance benefit of WX2 over a conventional RDBMS. Queries that took 3 days using a conventional RDBMS, complete in 20 minutes. This we have confirmed through our experience of looking for complex patterns in relationship data: queries that we were still running on our conventional database after 6 hours took 19 seconds using WX2.

Does this meet the criteria for BigDB?

When looking at Peter Gassner s criteria for BigDB we have the following conclusions:

- WX2 is a straightforward SQL database which, with the recent version 7, release supports Unicode data; it has a robust, standards-based SQL interface which can be extended by writing user functions but appears to be consistent with ORACLE SQL (though our DB2 SQL required some limited modifications to run directly against WX2)
- WX2 can be configured to be accessible through IBM InfoSphere Federation Server connectivity on a Windows Server using Windows/Kognitio ODBC drivers and on a client system running RedHat Linux using DataDirect ODBC drivers; we were able to access WX2 with a DB2 personality and using PASSTHRU via these technologies
- installation of WX2 took approximately an hour on a two blade system, each with 16 GB RAM and 280 GB of drive (running as VMWARE partitions on the separate blades)
- there is transaction control
- there are sound, effective utilities, for straightforward data load/unload and export.

On the other hand:

- there is still a need to execute a statistics capture utility, the results of which can then be used by the query optimiser; we noted a significant improvement in performance once statistics are captured
- the extent of the scalability of WX2 has yet to be proven by us — though we possess case studies of its use for very large databases (Kognitio claims that because of its architecture, provided the hardware is scaled through the purchase of sufficient — identical — blades and adding them to the blade centre then query times are unchanged as the volume of data grows; this is attributed to the MPP nature of the architecture).

However the clinching factor for us was value for money. WX2 licenses are priced per terabyte of data. This provides excellent value for money, particularly when compared with the software costs for the equivalent number of CPUs worth of conventional RDBMS licenses. In addition, the hardware platform provides exceptional value — requiring only a set of blades with local disks, and without the need for expensive I/O sub systems.

Should one's organization adopt such technologies?

Before introducing this form of new technology into an organization there are some key questions which I believe need addressing:

- is this just 'technology churn' (adopting the latest idea) or is this a long term investment?
- what, if any, are the hidden costs (say, additional training and support)?
- is there a positive investment return?
- what, if any, is the disruption to existing systems and processes
- is there a 'good enough' alternative from an existing vendor which can be rolled into an existing enterprise license?

From our own experience, there looks to be a strong positive case for consideration:

- for the specific problem class Diegesis faces, the performance differential between WX2 and existing RDBMS products is substantial
- the cost for equivalent licenses per CPU is considerably lower, and we obtain more from each CPU

- there is good integration into our existing data estate through ODBC and JDBC
- the integration story with other data analysis tools is strong
- WX2 is a mature technology (it has been in production for over 10 years).

Developing this economic dimension further, it is likely that WX2 will provide a positive Return on Investment (ROI) for the following reasons:

- reduced maintenance costs
- reduced database management time
- reduced engineering
- faster throughput demanding less investment in performance
- faster throughput producing more revenue (or cost saving processes being run)
- quicker speed to market of new data inputs
- cheaper license costs.

Indeed, we envision three parallel models for acquisition developing. These include where

- an organization only needs irregular use it can use a hosting vendor to process data (this is becoming increasingly popular)
- end users buy pre-configured appliances which provide a particularly economical approach (because the supplier can exploit its purchasing power to offer excellent overall value for money)
- the software can be delivered as 'normal', with the user installing and configuring the installation.

Management conclusion

The following key reflections can be drawn, especially in the light of Peter Gassner's call for BigDB:

- *it is now possible — even for organizations with limited budgets who need to exploit information more effectively — to acquire and manage the necessary technology far more cheaply than before, by using products such as WX2*
- *large organizations with substantial budgets can deliver results that were not possible within constraining time windows by using alternative technology and hence achieve substantial competitive advantage*
- *WX2 can integrate into existing data estates, with an organization still making use of existing reporting and data mining tools*
- *the benefits of something like WX2 are likely to increase as the complexity of the analysis increases*
- *WX2 will never replace conventional RDBMS products for transaction processing but may reduce substantially number of transaction processing licenses by trickle feeding changes in the TP systems into WX2 for analysis purposes*
- *within the wider picture it will be practical to expand the 'data estate' to provide integrated querying across all data sources (including unstructured content and semi-structured XML repositories).*

Our conclusions, therefore, are:

- *organizations with complex analysis requirements and currently using standard RDBMS technology are likely to see a substantial return on any investment in using products such as WX2; they should consider introducing these technologies*
- *the risk associated with a trial is minimal (the effort to run a pilot is small); if there are existing data and queries then the effort to install WX2, load the data and run the queries is almost trivial (and a trial could be run either in a hosted environment or on industry standard blade centres — which most interested organizations will already have or can rent)*
- *organizations not yet exploiting data mining techniques within their organization should consider WX2 as a starting point, to be supplemented initially perhaps with some 'roll your own' ETL and BI tools, safe in the knowledge that industry standard equivalents can be readily acquired and integrated into the environment*
- *these products, while not fully fulfilling Peter's Gassner's BigDB requirements, do go a substantial way towards satisfying them.*

Nick Denning
Diegesis
www.diegesis.com

Observations on what open systems mean today

Peter Bye

Management introduction

The usual motivation for adopting open systems is the belief that they increase business agility while reducing the cost of implementation and operation of applications. However, the terms 'open' and 'open system' are subject to a wide range of interpretations, plus they come with lots of baggage: 'what does 'open' mean?' and 'which systems are accepted as open?' are common questions. (UNIX-, Windows- and Linux-based systems have generally been regarded as open while mainframe-class systems have not, for example.)

In this analysis Peter Bye argues that the 'openness' of a system is a function of a combination of attributes and that openness is not related to any particular technology. This broader approach places more systems within the definition of open.

He argues that this is valid before going on to suggest that notions such as 'proprietary' or 'single source' -- terms frequently used when discussing open systems -- can confuse. He concludes, albeit (in his words) at the risk of mild digression, with some remarks on the relevance of software packaging and pricing, as raised by Charles Brett in 'The profound financial attraction of Non-Linear Licensed Software (N-LLS)' *(September 2009,* **INSIGHT-SPECTRA**).

Definitions, definitions and still more defining to do

The definitions of open systems have changed over the years. An early view was that an open system was one where the hardware was obtainable from more than one source. In particular, the availability of IBM plug-compatible hardware systems was regarded as making the S/360 and S/370 systems open as there were (then) alternative hardware sources to IBM (Amdahl, Fujitsu, etc.).

Later interpretations were oriented much more towards software — starting with operating systems (although the emergence of popular virtual environments, especially Java, has reduced the contribution of the relevance of operating systems to openness).

Vendor-neutral organizations tend to take a technology-independent view. The Open Group, for instance, defines an open system as "A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered Application Software:

- to be ported with minimal changes across a wide range of systems
- to interoperate with other applications on local and remote systems
- to interact with users in a style that facilitates user portability." (Note 1)

As might be expected, the World-Wide Web Consortium (W3C) takes an even broader view, focusing only on external interfaces: "An open system is one in which some producers and consumers are loosely connected or are not controlled by the same organization. The internet is a good example of an open system." (Note 2)

What emerges from these definitions is that systems are not either open or not open. Rather, openness derives from a spectrum of attributes. Evaluating any particular system, or class of systems, therefore, requires ascertaining what open attributes a given system possesses; only then can one ascertain where a system sits on the spectrum of openness.

Assessing openness

In assessing the openness of any particular system, I have often found it useful to view assess the problem from four perspectives divided into two groups. These try to cover the key attributes (see also Figure 1):

- the system on its own, with a network of end users but not part of a distributed environment:
  - internal interfaces, which include application environments, file and database access, storage subsystems and communications
  - application development tools and the application process that can be used

- the system as part of a wider distributed environment, where it collaborates with others, for example in a service oriented architecture:
  - external interfaces, including distributed application servers, middleware and distributed transactions
  - systems management: can the system be managed using the technologies and products able to manage other common systems, or at least interface to them?

The mainframe is as open, or more so than alternatives

An analysis of mainframe-class systems, based on the above approach, shows that they are far more open than many people believe. (Those who have never worked on them often seem to think that there has been no development since the 1960s and 1970s.) Today mainframes typically support a variety of application environments, including:

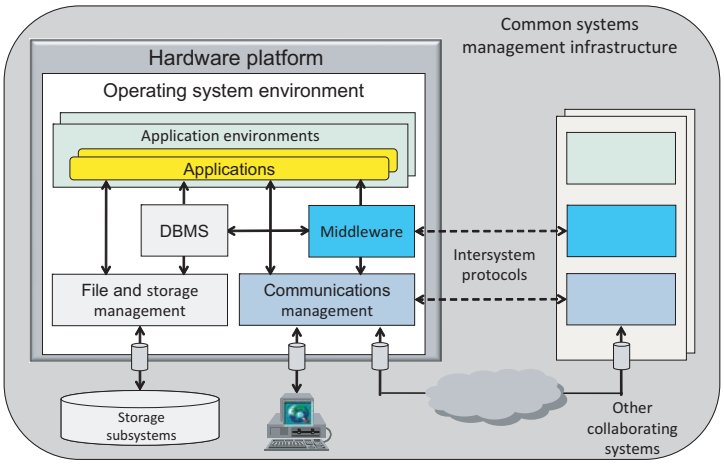

Figure 1: Components when considering openness

- Java EE
- Open Group DTP
- industry-standard database and file access methods
- a rich range of middleware.

They are also manageable by using common tools along with other systems in a data center. In addition, applications and other software, including open source, can be ported in with little or no difficulty.

Thus, there is no reason to rule out mainframes because of any supposed lack of openness. Indeed, mainframes also bring many advantages in large scale environments with intensive transaction processing. They can reliably handle multiple, different applications at high load factors, approaching 100%, while still running efficiently.

In addition, given current concerns about security, mainframes protect valuable data from compromise. Evidence for this can be found in a NIST database, which records security vulnerabilities (http://nvd.nist.gov). As of 9th June 2009, the number of vulnerabilities reported in the database were:

- 1560 (Windows)
- 1263 (Linux)
- 181 (UNIX)
- 3 for mainframes (showing just 2 for IBM's zSeries, and 1 for the Unisys MCP — in which no data was compromised).

Many enterprises still run large, critical systems today on mainframes. A reason typically proffered by those thinking of replacing these is that they are not open, and that they are therefore inflexible and difficult to work with. However, as the above analysis suggests, mainframes are well along the openness spectrum and can be integrated with new capabilities in some form of service architecture. Integration is frequently much easier and cheaper option than replacing mainframes. In fact, mainframe replacement projects almost always turn out to be far more expensive and difficult than expected, all too often never recovering their cost.

Proprietary and single source

The use of terms such as 'proprietary' and 'single source' complicates discussions about openness. Proprietary is frequently interpreted as bad and therefore to be avoided. But the majority of products are proprietary in the sense that they have just one source. What matters is that they have open attributes, such as standard interfaces.

Similarly, two different products, each proprietary to its supplier, may have identical open standard attributes; J2EE/Java EE application servers are examples. They can be differentiated on price, performance, and often on features that are not standardized and

so depend on specific implementations, for instance resilience. But it would make no sense to dismiss them as being proprietary.

Thoughts on Non-Linear Licensed Software (N-LLS)

I finally address some of the points made by Mr. Brett. Choice of software products is one of the reasons advanced for open systems. However, with choice comes the problem of software integration. Because infrastructure software stretches, as Mr. Brett puts it, from the "metal through to below the business application", it is essential to ensure that all the components fit well together and perform to the required levels, or the applications will not deliver what is required by the business. This is a non-trivial integration task, which has to be repeated, at least in the form of a regression test, with each release of new components.

An organization using an infrastructure stack in software can take two broad approaches to its integration. It can undertake the integration itself, which requires maintaining a team with suitably diverse skills for the purpose. Or it can have someone else do it. The alternatives for the latter are by outsourcing the whole operation or working with an organization that specializes in pre-packaging the software.

IBM, Microsoft and Oracle among others are able to perform the integration because they provide many or all of the component parts, from the operating system upwards. And incidentally, owning all the components allows substantial optimization of the software and provides just one place to go for support.

However, Mr. Brett points out that it may not be not in the interests of these companies to adjust the licensing structure to an N-LLS model rather than charging by the number of processors or cores in the host platform. (It may be noted in passing that there is also the complication that users do not always choose to use the full software stack, for example using Windows with a J2EE application server from JBoss and a database from Oracle.) He goes on to suggest that he expects new companies specializing in packaging stacks of software on an N-LLS model to appear.

My own experiences tends to agree with his view. For example, the approach taken by Unisys with its ClearPath systems is interesting and, I believe unique. The systems come with a comprehensive software stack, integrated and optimized for each release. There are various pricing options, of which one increasingly popular choice is a pay-for-use model using metering technology.

The systems are supplied with full power but the user pays for the power consumed; processor power is treated as a utility. This is directly analogous to electricity metering but instead of kW hours as the unit, the user pays for 'MIPS/time' units. However, the full power is always available and so can cope with any peaks and surges in demand as well as accelerating processes. (More traditional models rely on supplying sufficient power for peaks even though the average consumption may be much lower.)

Management conclusion

Mr. Bye's point is that open systems are changing. Most sytems are relatively open, certainly by comparison with 10 or more years ago. Cloud computing is producing still more openness and N-LLS (if adopted as Mr. Brett intimates) will only extend openness further.

The good news is that all this build on what has successfully worked before. All those arguments about 'openness' in the past were 'not for nothing'.

Notes/references

(1) http://www.opengroup.org/architecture/togaf8-doc/arch/chap36.html
(2) http://www.w3.org/2001/tag/doc/versioning)

Peter Bye
Independent Consultant

Value-base case study
Managing an MQSeries environment at USBank

Management introduction

USBank is headquartered in Minneapolis and is #6 in the US (as measured by assets), USBank has some 50,000 employees. David Corbett is a middleware architect in the Bank. He focuses on multiple forms of middleware: this includes CICS and MQSeries plus TIBCO messaging (USBank, in common with most large organizations has no shortage of different forms of middleware installed).

In this case study, Mr. Corbett discusses the development and operational issues in USBank that arise with MQSeries and how these are managed. He describes issues arising between operations and development and explains how and why USBank introduced Avada Software's Infrared360 solution as a way to improve both the Banks's application development and operational environments.

The business problem

We do not have, at USBank, a really complex MQSeries environment (I know of ones that are much more complex). Our MQSeries network supports traffic to and from the mainframes but there is also traffic over the MQSeries network that goes between non-mainframe (distributed) systems. MQseries is, therefore, an integral part of how we make applications work together within the Bank. As such we need access, for all sorts of valid reasons, to see what is happening in the queuing infrastructure — but with strict controls operating on what can be seen by whom (for security and confidentiality reasons).

A further consideration is that we are a bank. We are highly regulated (by the Office for the Control of the Currency) and there is always great concern about traceability and logging. It is vital, both for own business integrity but also to satisfy regulatory and compliance requirements, that we know — and can show — what goes on where, why and how.

This has some interesting implications. Take, for example, the concept of most middleware tools. They 'demand' a customer have a designated administrator — who will have the necessary privileges to operate that software and, with these privileges, complete control. Banks struggle with this because those that know and own expertise about how to use particular tools in depth are often in development. But developers generally (and correctly) should not have the rights that (say) operations must possess.

In effect there can be a conflict of interest here. This needs most carefully to be managed in order that regulatory and compliance requirements are fully satisfied. There must be evidence that the requirements are satisfied. For example at USBank we have had to create emergency access IDs for certain types of operation; these are held by security. If we, in development, are asked to solve a problem which we (in development) would not normally be allowed to do, we have to request an emergency ID from security — and the subsequent actions using this ID are tightly monitored and logged.

In addition, a background factor was driving us to look for new approaches. This was our need to consolidate our MQSeries network which had evolved over time. We did not have what I might call an 'MQ Farm'; for example, nearly every application using MQSeries had tended to purchase its own dedicated MQSeries Managers. This was expensive. In addition, MQSeries was an obvious candidate for rationalization and restructuring, if only to make its (MQSeries') usage more coherent and manageable.

Development and production in an MQSeries world

MQSeries is an interesting piece of middleware. Normally, in most production environments, operations can run with what works. This is not so simple with MQSeries, which often requires a good understanding of the related and associated applications. This means that application development often becomes an integral part of ensuring an MQSeries network continues to function. On this basis, MQSeries can be regarded as something rather different to much middleware.

Yet it is difficult in a large organization to step up to the plate to take ownership of how MQSeries works across the enterprise. Putting everything together for

operations is dependant on how much operations is prepared to take ownership of MQSeries. Part of the issue is where MQSeries is running. For example our mainframe MQSeries implementations — and, for that matter our Tandem one — function in a normal, big computing environment: this is proven. But our instances of MQSeries running on our OS/400s, AIX, HP-UX and Windows environments often do not share the same operational disciplines as we have for our mainframe/Tandem environments — even though they do 'talk' to other (or both) of these larger systems.

What many people do not wholly understand is that MQSeries, once it is running, is extremely stable. But one (unintended) consequence is that, once running and stable, people forget what they have done and why. It may seems strange but one area where Avada Software's Infrared360 assists us is with 'coping' with difficulties that arise when something breaks (which everything does at some point) and when institutional memory cannot recall what should happen. Avada's software gives USBank that extra degree of depth and understanding that overcomes the difficulties which arise when people cannot remember. It institutionalizes knowledge about key areas of our middleware environment.

What went before and what we needed

Before we purchased the Infrared360 product from Avada Software (Avada) we had a tool for developers and with which a developer could go and look at his or her queues (and only their queues) to see what was happening. It could also do some limited manipulation, but this was not sufficient for us.

Why was this important and even a business issue? MQSeries is middleware that is essentially invisible but is also tightly integrated to the applications which it connects via its messages and queues. As we have found out the hard way, it is often very difficult for production/operations staff to understand how an MQSeries network runs or what are the application implications. In addition, production did not possess any tools of its own to make this easier.

The result was that, when a production problem occurred that was MQseries-related, the operations people all too often could not resolve whatever was wrong. Its solution was to call development to seek assistance in fixing the problem.

In essence what we were looking for was a solution which could:

- monitor the MQSeries network
- enable those who had to manage production to do this without needing specific MQSeries expertise
- avoid the need for installing agents
- authenticate users via our LDAP tree (for security, so that we could know actions were only performed by those authorized)
- be browser-based (which was very important to me personally — because it is a commonly understood presentation 'mechanism' and avoids having to install clients everywhere)
- reduce the load, thereby, on development
- support our introduction and management of a farm of MQSeries servers.

Introducing Avada's Infrared360

We heard about Avada almost by accident, from our Tandem tech. services support person. It then emerged that I knew one of the founders and developers from when he worked at a previous software company. As we were already looking at other software (specifically MQSoftware and BMC's Patrol), we included Avada in the evaluation we had started.

In the assessment what became clear was that, although the solutions from MQSoftware and from BMC had good features, their biggest drawback was that both would require us to run agents everywhere (which would add a complexity we did not want). In addition, both seemed to demand we have somebody almost full time just to manage their tools. This did not make much sense to us; nor was it attractive.

A further factor pushing us towards Avada was that its solution was less expensive. But most of all Avada appeared to meet more of the requirements that I described above than its competitors. The final 'plus' was it offered a browser interface (removing that need for clients) — which was unique amongst these three possible solutions. Indeed, if I look back, I think the one factor that would have prevented the choice of Avada would have been if there had not been LDAP integration for security (all three offered this).

Avada's software in practice

What we are finding is that the Avada approach does

enable us to be more flexible. We can write flexible rules, of increasing complexity, which satisfy our needs. (This is one of those areas where it is often not easy to be sure you have bought correctly — until you are deep into using a tool. The evidence, thus far, is good.)

The browser-based interface is as we want, though it is not as intuitive as I would have liked. The answer is, however, reasonably simple: we offer more training and people have to learn how to work the Avada product through it. Once they are trained, it works and user friendliness certainly comes with greater familiarity.

Rules in Avada are XML-based. This has various advantages. We understand XML. This means that, if we need to, we can go in and tweak the rules though editing the XML — which is relatively painless.

When it comes to tying various aspects together —— for example, the fact that a channel is not running and the transmission queue has undelivered messages yet there is no reader for that queue even if the channel was running or should this queue manager being running at this time of day — Avada has deep capabilities. Addressing this sort of problem in an automated way is not easy at any time. But Avada makes it much easier in such 'compound' situations. I expect that, as we progress further, we will use more and more of these sorts of capabilities.

Indeed, we have not yet been able to come up with any rules scenarios that we think Avada would not be able to support. Equally, the capability to build a rule and re-use it (without having to redevelop it) is attractive.

Besides doing what we want, Avada's software possesses a statistical database (operating behind the scenes). We have not really used this yet. But we can see possibilities. If we choose we can exploit the statistics that are generated (if you turn on the MQSeries statistics function). This can then be used for analysis, for identifying weak points or bottlenecks (or whatever). This has the potential to be operationally useful in the future.

Fundamentally, however, what we are building with Avada is an MQSeries environment where the Bank obtains pre-emptive notification of problems. This is quite different to the reactive environment with which we had had to work previously — and much sounder as well as more reliable. Now we do not have to wait for people in the Bank to call us to say 'we are not receiving our messages' as the first indication of a problem. We are fixing the problem before they can even call us — because we already know.

The net result is that application development, where I am a middleware architect, provides operational support to production for non mainframe/Tandem functioning of MQSeries. To do this Avada's Infrared360 product has become significant in our effectiveness — so that the development staff can see their queues on mainframes as well as distributed systems. This is not so much from the monitoring perspective as the ability to see what is happening when the problems occur.

Lessons learned and best practices

As a regulated financial institution we had and have particular security requirements. I would say from our own experience that even if you do not have such obligations it is both desirable and a good practice to sit down and formally lay out the groups and roles that you want to assign — before you start building-out your application accessibility.

Equally you should spend time working through what are the generic rules you should apply, and then create these. Doing these early will save much time later. Even though this did take us longer than I expected to come up with an acceptable model for the groups it was part of learning to use our new tool.

For controlling access in the way we wanted we needed to do the filtering at the group level, which was pretty much self-evident. Defining the groups — as well as which MQSeries queue managers and what could be seen — was trickier. It did force us to think clearly, which was particularly necessary for system queues. Now done, the time invested was worth it.

One issue we did run into concerned queue managers. This had two impacts for us. The first related to how queue managers are used. Most developers tend to think more in terms of having their own queue managers, rather than in keeping these down in number. This was a good reason for pursuing the 'MQ Farm' approach. But there was another side to this. As part of our preparedness we do disaster recovery testing every quarter. This means we have to have all the necessary disaster recovery infrastructure in place. We still want to use Avada's software for

this. If I have the queue managers already defined, we have to possess additional queue managers. If we choose not to do this we create a major configuration problem for ourselves on disaster recovery testing day — which is equally undesirable.

But perhaps the biggest lesson we learned was, and is, the value of laying out the architecture of your MQSeries infrastructure — especially differentiating between development, testing and production - before you start applying and exploiting a solution as flexible as Infrared360. The temptation is to rush in; waiting, thinking before applying pays dividends.

One other indicator of success is worth mentioning. Our security, operations and internal auditors are much happier than before. Avada's software enables us to satisfy their needs in ways that do not hinder — and even do assist — developers do their jobs. This is a win-win — for keeping compliance and internal auditors happy in banking matters.

Management conclusion

USBank is typical of many large organizations — with a disparate middleware infrastructure. As Mr. Corbett illustrates, where USBank is different is that it has looked at its MQSeries environment from a modern banking and compliance perspective and sought to consolidate and improve.

The key lies with how MQSeries works in practice in most organizations. Broadly it is so reliable it is forgotten about. In one sense this is good — not much support energy has to be expended. But in other ways this leads to downstream problems — like in how development and operations should work together.

In selecting Avada Software's Infrared360 product, USBank introduced a flexible solution that meets the internal audit and compliance requirements while also addressing the needs of both operations and development.

Value-based case study
An event analysis protype leads to evidence-based medicine — and the potential to predict heart attacks

Management introduction

When Oracle, Utah Health Sciences Center (UUHSC) and the University of Coimbra decided to investigate if patient-related issues raised by doctors could be addressed using new IT approaches, this was an unusual step:

- *most doctors do not understand the technicalities of IT or data processing*
- *most IT professionals are not trained as doctors.*

By good fortune a combination of doctor and IT researcher (a daughter/father team) crossed these boundaries (and mutual ignorance) and started to explore a framework for researching how analysis of medical data might be automated and thus turn data into evidence for doctors to improve their treatment of patients. Using anonymous historical data gathered from UUHSC's surgical ICUs, a prototype was constructed to investigate whether using IT to analyze medical events might provide significant insights into:

- *what medical data exists that might be analyzed*
- *whether event processing technology (particularly that related to the value of state-based event processing) was relevant*
- *what sort of complex, highly customizable, extensible application might need to be developed (which avoid the static, domain specific procedural code that is prevalent today)*
- *how to define (in order then to deliver) 'evidence based medicine'.*

Put differently, the prototype described below investigated whether there was the potential to undertake analysis and thereby significantly improve the clinical care doctors provide to patients. The prototype succeeded on at least three levels:

- *there is sufficient evidence, obtained from the analysis that heart attacks can be predicted 24 hours in advance*
- *the information technology worked as anticipated*
- *formal clinical trials will shortly commence.*

The background

The prototype was tasked with examining the potential for medicine to exploit existing diagnostic events by using the analytical abilities inherent in IT to improve predictive capabilities plus use evidence to support medicine in ways that go beyond current medical record keeping. It involved:

- the University of Utah Health Sciences Center, with anonymous data and medical expertise supplied from its Surgical Intensive Care Units (ICUs)
- Oracle, which provided the analytical and event processing technologies
- the University of Coimbra in Portugal, which contributed the analytical support.

While what is described below applies to post facto research (looking at past, rather than current, patient data), the results were of such significance that formal clinical trials are scheduled to start in 2010. Unlike many clinical trials, starting should be straightforward because no alteration of, or adjustment to, a patient's current treatment regime is required: the clinical trials will subject a number of patient's existing clinical data to analysis. These analytical results will then become available to the doctor responsible, who will remain the person in charge of the patient's clinical care.

The current state of play

It may seem strange but IT supports non-patient care — such as billing and accounting — far better in the USA than it provides clinical support for physicians and their patients:

- 4% of US physicians report having an extensive, functional electronic records system
- 13% of US physicians report using a basic electronic system
- 1.5% of US hospitals have a comprehensive electronic records system
- 7.6% of US hospitals have a basic system.

According to two recent articles in the New England Journal of Medicine, less than 14% of US physicians and less than 8% of US hospitals report using even a basic electronic medical records systems. A different (rather alarming) way of putting this is that most hospitals are 100% equipped with IT for commercial functions (billing), but are far less well equipped to apply IT for direct patient care.

In effect, even though medicine already possesses large numbers of sensors generating data or events, that data is largely 'lost', with the quite possible effect that evidence (data analyzed to create information) for doctors help treat patients is either ignored or, worse still, not even known about.

Alert fatigue and loss of focus

To understand how this can happen it is necessary (briefly) to describe some parts of the environment. In most Intensive Care Units (ICUs) there is a wealth of data created each second. Just think of all those monitoring devices — for heart activity, blood pressure, respiration and multiple other functions. Each device measures something and produces data points (events), and floods of data.

ICUs are, therefore, data-rich environments. However, whether for doctors or for traditional IT — at least to date — that data's existence has rarely been recognized. The result is an analysis-poor environment.

What happens to this data? Yes, there are screens and displays so that medical practitioners can see what is happening at any given moment. Yes, printouts are available or can be made. Yes, some data may have on-going analysis — for example watching for catastrophic events (such as an ECG issuing an alert if a heart stops or falls beneath a set parameter).

Doctors, however, do not think in terms of numbers (data). Instead they tend to think about how good or bad — relatively — is a medical situation. This means that doctors swiftly use (and then largely forget) the specific data after classifying a patient's situation. It is this classification — serious, critical, etc. — which is remembered and which motivates actions.

Yet one of the biggest problems in hospitals, and ICUs in particular, is 'alert fatigue'. Too often alerts are issued when no alert is medically needed. For example, an elderly person's heart beat will likely be dangerous if it goes over 150bpm; yet this is not necessarily so for a newly-born baby. If no distinction is established an alert may be issued when it is not necessary.

With so many devices, medical practitioners are deluged with machine-generated alerts, all too often indiscriminately issued for basic incidents (such as a value going 'out of range' — which is not uncommon for one or two readings in an electrically noisy hospital environment) rather than for the occurrance of a known problem based on the assembly of evidence. Indeed, to make matters worse, if a patient has a severe renal problem, it is quite likely that all doctors associated with that patient or ICU will be alerted, rather just the renal specialist. It is wasteful that perhaps the cardiologist or urologist are also immediately alerted.

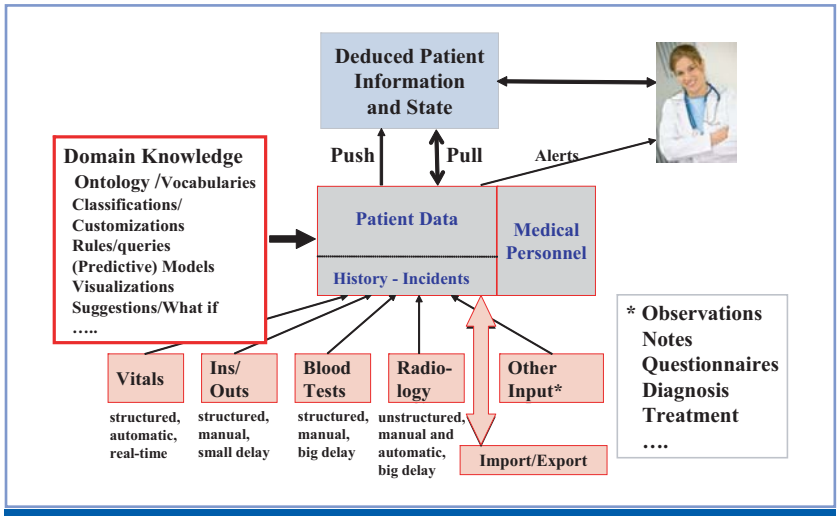

Figure 1: Patient care overview

One consequence, then, of so many alerts which are not classified in some way is 'alert fatigue'. Medical practitioners do not know whether an alert is more or less serious for his or her specialty.

Further, discussions with UUHSC medical staff revealed that many of its doctors did not feel they had control of the the very systems that, in theory, were there to support their skills. Minimal customization was pratical, even to accommodate specialty expertise or to try out new treatment approaches.

What are doctors looking for?

Doctors would like to possess more control, to feel that the medical systems by the bed and in the lab. are there to support them (the practitioners) to treat the patient — rather than just surround the patient with expensive technology. Doctors at the UUHSC wanted:

- support for extracting evidence from the data generated
- unobstructed access to all patient data
- unobstructed support for their inquiries and analysis
- to be made aware of their patient's adverse conditions in a timely yet relevant fashion (as soon as these happened along with alerts based on a pre-described level of urgency and with information about what threshold had been crossed).

In reflecting on these, the need for integration was all too apparent. To access all patient data means that integration must occur. From a doctor's perspective, the complexity of the medical systems environment is captured in Figure 1. Or, as Dr Kimball at UUHSC has described it: "I'm sitting on a mountain of data hidden behind procedural code".

Minimal device and IT integration

Worse still, most of the diagnostic systems (whether online and/or in laboratories) lack integration of the data generated for each patient. Each device or system too often is standalone. Systems, or the data being generated, are not combined so that the medical practitioner can obtain a whole view for a patient. This is made worse by delays. Some tests – for example blood analysis — take hours before the results are available. Even when the results are ready, it is the doctor who integrates the results — rather than have all results integrated (and analyzed and recorded) for him or her then to take decisions.

Integration matters in other ways. Consider clinical

documentation. Whether for immediate analysis or for assisting future consultations, bringing together all the data together for an individual patient is highly desirable for treatment purposes. It is also necessary for tracking what medical initiatives work on whom and why. Furthermore, on an institutional basis, the lack of integration at (say) the ICU or hospital facility level means that decision support is often not available and, therefore, outcome tracking is not automated. For example, in the prototype, it was necessary to enter data and integrate patient data manually before the analysis could start.

The technology approach taken

The basis of the prototype used Oracle's Streams Processing.

Events are captured as 'facts' – and there is even a good correlation (see Figure 2) between traditional and new technologies – as well as the 'participants':

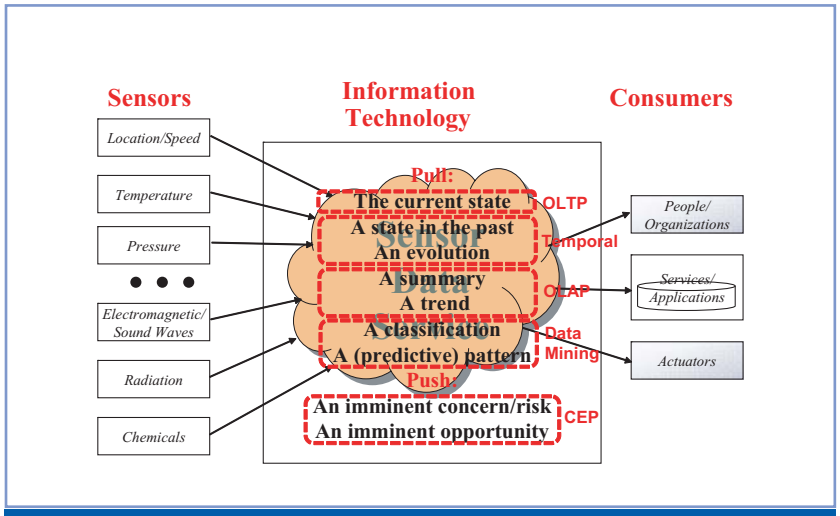

Figure 2: State-based event processing

- the medical sensors
- the events processing performed in Streams Processing, plus an event journal and an event warehouse
- the consumers (the doctors).

The Oracle technologies used were based around 11g and included:

- Total Recall (for transparent versioning of records as well as access to historical data)
- Continuous Query Notification (CQN); this provides the notification system (including notification on change of result set) plus support for standard queries plus destination (using SQL, XML, SPARQL/OWL and with extensibility)
- Rules Manager (RM); this rules engine enables the pattern matching, complex alerting, etc.
- Oracle Data Mining (ODM); this provides the mining engine and is embedded in the database (it also allows for scoring directly from database data).

What happened in the prototype?

Looking back over the work performed in the prototype in April-June, 2009, various different conclusions were drawn. Some of these are obvious, in hindsight, while others hold out the potential for real improvements in how evidence can be generated for use by doctors to treat their patients. The conclusions can be summarized as:

- ontologies matter, and fortunately the medical world is strong here
- medical classification simplifies the creation of rules that can be applied broadly (or narrowly)
- integration improves documentation standards and record keeping
- medical data from the past can be used to predict what may happen in a patient's near future (the best example is that blood chemistry changes were observed to occurr some 24 hours before a heart attack).

The medical world is good at knowing what it means. It possesses a common ontology. Each doctor knows what blood pressure or heart rate means. This represents consistency. Possessing an accepted ontology made the analysis and then the building of rules and classifications much easier than expected (but see Note 1).

Being able to classify has real advantages. It means that the rules can be made generic — rather than be

created for individual patients. In turn this aids querying. The prototype also showed how much easier it is to ask for a list of all patients conforming to a particular classification (say, low blood pressure). Such flexible querying is not possible in most hospitals today. Indeed, this was one of the most interesting extensions of capabilities demonstrated by the prototype.

One of the imperatives in the medical world is that everything must be documented first and precisely — before anyone does anything. Physicians must be able to follow what another doctor did and, preferably, see the data and reasons why particular actions were taken and treatments administered. Without this, a patient may be placed in peril — through ignorance of what and why had happened previously (rather than through negligence). When there is an automated process that integrates data this creates evidence, with the 'byproduct' that medical records are updated and complete – making it much easier for doctors subsequently to see what happened and why.

A different aspect that emerged was about the difference between past history data and what predictive capabilities might offer. All measurements are data points about what has happened (or is happening). But in the past this did not look forward, to predict, which is desirable.

For example, the prototype found that patients' blood chemistry is routinely analyzed in ICUs. Yet this is not often a source of evidence to which most doctors turn (unless they have a specific question) — not least because blood chemistry results take time to arrive: they represent a measurement point in thepast.

Yet, when the prototype took history-based data, integrated it and then performed analyses it became apparent that in most heart attack patients there is a change in the blood chemistry some 24 hours before an actual heart attack.

Nobody had seen this before. Why? Because past blood chemistry data was rarely integrated into the evidence that a doctor used to review a patient's current changing condition. Yet, by integrating and automating the data analysis it was possible for the prototype to generate evidence and then 'warn hours in advance' that a patient had undergone blood chemistry changes that were indicative of a forthcoming cardiac incident.

The significance of rules

A related aspect that emerged in the prototype is that it is important to doctors not only that they be able to see what has been done by others (and why) but that they can apply different rules if the individual medical practitioner thinks this is applicable. Rules, therefore, have to be flexible enough to be able to apply to different conditions, to different patients and even be used by different doctors — and all must be recorded. Doctors need, therefore, to be able to customize rules as they feel necessary for particular patient situation — without losing the documented picture.

This has additional implications. Today's medical 'rule base' can readily be classified. Once accomplished this becomes an asset that can be analyzed (for example, outcome analysis). In addition it can suggest additional treatment options or choices — ones that have been tried by other docters. In effect, doctors can look for what treatments have worked in particular circumstances in the past, or what additional tests and analyses are suggested by a particular patient condition.

This would be a huge step forwards because it would enable doctors to use what they cannot now see. Doctors really do not know what is in the raw data. Often they do not even see it (the raw data), unless they specifically ask for it. Today, relevant data is hidden by the complexity of the hospital as well as IT environment. It is not that doctors are stupid (even if some do seem to use arrogance to hide ignorance).

What the prototype showed was that, by giving doctors evidence derived from analyzing raw measurement data, it is possible to change the diagnostic as well as treatment approach. As such, event processing demonstrated that it can be a meaningful contributor to developing information and good procedures.

Moving to clinical trials

The objective of the prototype was to discover if 'evidence–based medicine' could improve current medical practice. The results were positive.

The next step is take what was initially an interesting concept, strengthen the prototype — to automate more and improve the interface for the doctors — and move to clinical trials. For these trials some 50% of patients will be participants whose doctors use the automated analyses while 50% will not. By comparing the outcomes it will be possible to see if this

approach is valid. Indeed, as part of the analysis of the outcomes it will even be possible to go back over the data of those who were not participants and see if there were any of the same indicators or predictions that might have been picked up, had those patients been full participants.

Could there be harm to a patient? No. Because this whole approach is based around providing evidence, and predictions; it is not about treatment. The doctor remains the decision maker who decides what treatment to offer.

Nevertheless, not everything has been put together yet. The integration and related infrastructure from the prototype is there, but this will need to be tweaked and reworked for the trials in order to create a coherent solution.

Yet what is clear is that there is more than enough data to support this approach even if that data has to be integrated into some form of store/medical records against which automated analyses can be run (and action rules written) which search for known past patterns which are predictors of bad (or good) changes in a patient.

Management conclusion

This prototype focused on the creation of evidence — essentially analyzing data (events) to derive that evidence. It moved beyond the hard coded, separatist, approaches of medical IT today — which is difficult to change and also has the failing that, if something is not coded, then various aspects cannot even become evidence. As such this prototype represented a rethinking of the way data and evidence are used in a medical environment.

What Oracle. UUHSC and the University of Coimbra — whose participants never physically met during the prototype — demonstrated was that the processing of events has a potential place to play in the treatment of patients. It also discovered much about how and why the medical profession does what it does, its various strengths and weaknesses and what improvements should further be investigated.

Nevertheless, it is also worth remembering that the 'medical events' are not the end game. Creating evidence is. Roll on the clinical trials.

Notes:

(1) That is not to say that everything in a medical ontology is universal. In the US there is a simple diagram that is used to convey multiple pieces of information to a doctor. All US-trained doctors recognize it. But to a German or UK-trained doctor, this same chart is probably meaningless.

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

All rights reserved; INSIGHT-SPECTRA may be reproduced for individual use. However, bulk reproduction (more than 5 reproductions -- whether by electronic means or in print) requires prior written permission from the Publisher, C3B Consulting Ltd.
© 2010 C3B Consulting Ltd.