

# International BANKING SYSTEMS

THE JOURNAL OF BACK OFFICE OPERATIONS

MAY 2007 / ISSUE 16.8

ANALYSIS: APPLICATION LIFECYCLE MANAGEMENT

IBS MAY 2007

## Good housekeeping

Rather like keeping your bedroom tidy as a teenager, tasks requiring an orderly approach do not come easily to some people. But when the bedroom is exchanged for the boardroom, those tasks become rather more challenging and important. In the IT zone, Application Lifecycle Management may be the answer. Tom Alford investigates

Without a strong sense of purpose and a fastidious approach to monitoring and control, keeping tabs on every IT change can be difficult in the extreme. And yet some businesses continue to promote chaos over order and barely have a grasp on who is doing what, where and how.

It would be fair to assume that a technology package exists that is able to keep matters under control. Unfortunately, given the diversity of processes within the financial markets environment, anyone seeking a one-size-fits-all automated clean-up package will be disappointed, says David Upton, managing director of UK-based consultancy and software developer, DBFS. He suggests that those who failed to learn where everything goes in their formative years are rather living on borrowed time if they carry their disorderliness into the ultra-competitive business environment.

Those that are aware of the importance of keeping processes under control will perhaps also be aware of the practice of Application Lifecycle Management (ALM). The problem is that ALM is often seen as an isolated concept dealing solely with IT development, which to an extent is what it is.

But whilst IT development lies at the heart of the matter, Upton notes there is

so much more to ALM. The pre- and post-development stages also need control. Solutions exist to administer these aspects, from the early days of project management to licensing and beyond. Problem number two with ALM, however, means that at the moment it is 'pretty fragmented'. There is no single across-the-board solution on the market yet. In fact, notes Upton, 'I don't



think you could meet all ALM requirements from a single piece of software', simply because every business has different IT needs and approaches.

But the lack of a single out-of-the-box solution is no reason why an ALM pro-

gramme cannot be set up. Of course, it depends on how seriously a business takes IT control, but the current general lack of focus in this area needs fixing if progress is to be made. At the company level, the only way this can be achieved is through one individual or office having enterprise-wide oversight. But many firms have globalised their development resources. This presents issues of standardisation and coherence where a business spreads itself across multiple sites, countries and time zones. 'In this situation, management oversight is critical to ensure co-ordination,' states Tieren Zhou, president of California-based ALM software vendor, TechExcel. 'The level of integration achieved by the entire development process ultimately defines the scale of a project's success.'

In any case, Upton notes that 'few companies are that well organised' but nonetheless believes the role should be adopted and run by committee. It is important to achieve the right balance of power though – of those firms that do have something approaching an overview, the tendency is to focus on one area such as architecture, project management, operations or development. To counteract this, any management committee will be made up from a mix of personnel including the business

executive, the project office, operations and support staff.

Because the real starting point of the applications lifecycle comes well before the development and ends way after, Upton often uses the phrase 'from cradle-to-grave' to describe the relationship. Starting from scratch, it can be seen that any IT requirement can be traced right back to the bank's original business needs. These typically take their cue from commercial or regulatory pressures. The need then has to be represented as part of a software solution, each aspect of which must have a pre-defined technical specification. The specs could be resolved either by buying or building, or a combination of both.

The result is that companies end up with so many pieces of software that have become so important to the processes which underpin the services or products they are offering, that in a lot of cases many businesses simply couldn't operate without them.

Upton sees that many companies are only just waking up to how dependent they are on their software. And whilst many just about manage their software in isolated

are 'extraordinary'. Maintaining control is becoming increasingly difficult for everyone involved. This rather plays into the hands of the argument for having a clear ALM strategy: it could well offer a lifeline in the rising tide of software.

No business is too small to have a policy. 'Any organisation that has a mission-critical business process should use it,' insists Upton. 'The question is, how formalised the discipline should become, and to what extent it is just common sense.' Indeed, how much thought is given to the matter can depend on the complexity, size, and importance of the business process in question.

In reality, even the smallest of business software users are to a degree already engaged with ALM, albeit perhaps unwittingly. When a company submits to upgrades from its software vendors, or has to buy new software as a result of buying new hardware, or even instigates a training programme as a consequence of licensing a new package, a software user is entering into the lifecycle process. The problem is that many programmes are short-lived. Out of sight, out of mind.

this covered by the single development tool?

Consider that a company may buy a product that has a 90 per cent fit, with the remainder coming in as an in-house development. What is the definition of the application under management, asks Upton. Is it just the ten per cent in-house development? How is this separated from the body of the package in terms of ALM, and how is the 90 per cent managed?

Indeed, the reality is that a bank is looking for flexibility, scalability and resilience. If it is to achieve them it needs to take a grand view of every single application it has, how these might be changed, who needs to make the changes, when and how the change should be made, and what impact the changes will have elsewhere.

The latter point is particularly relevant to the implementation of a third party solution into a typically complex environment. The vendor's sales staff may assure the client that their application or even their upgrade of an existing package will slot straight in, but their software is unlikely to have been developed in a mirror image of the system it is heading for. Additions or alterations to a live system may well make unexpected systemic demands further down the line.

Of course, once up and running, third party solutions require a licence. Optimising this aspect in itself can prove to be a real minefield. Paying for redundant licences is just inefficient, but failure to manage licensing can also be limiting on an organisation, notes Upton. Some vendors are 'particularly diligent' in following up licence abuses – inadvertent or otherwise – and may impose penalties upon offenders. More importantly perhaps, insufficient coverage can see vital users locked out of mission-critical systems. As Upton states, 'You really do need processes and procedures that manage software that somebody else has developed'.

The skill therefore is in knowing where and how something fits in relation to everything else. This is where change management software, impact analysis tools and a support database can help. The latter is a kind of inventory tool that a lot of businesses developed around the time of Y2K to help remediate their hitherto uncatalogued systems. One of the inherent problems here though is the work that is required to

**'Traditionally, businesses that have had good applications maintenance and infrastructure programmes have just looked on it as good housekeeping'**

David Upton, DBFS

pockets, he believes that the only way to gain real insight and control is by adopting an holistic ALM approach. Just 'getting in there and doing it' unilaterally is not good enough.

The holistic approach is a key part of the development of security for an organisation in terms of knowing where and what it has in place, how it works, and what the impact of change is on its various systems. It provides an important view into best practices, processes and procedures for developing, implementing, supporting and maintaining mission-critical applications – from cradle to grave.

In defence of the less well organised, it is easy to criticise when looking in on a company from the outside. 'The devil is in the detail,' agrees Upton. The complexity and increasing sophistication of some applications, the number and variation of clients and the demands of cross-border operation

However, Upton has noticed increasing numbers of larger organisations developing some kind of formal ALM policy, and adopting best-of-breed management solutions for each aspect. They are taking on board discrete packages to help with management processes covering issues such as project management, testing, QA and licensing as well as the more obvious application development tools.

For now then, as a fractured process, most firms start further down the line than perhaps they ought by controlling the obvious – the application development lifecycle. For this, a number of solutions exist.

Typically, a request for a new piece of in-house software requires specification, writing, testing, QA, deployment, training and then support, before following the cycle repeatedly until the solution is decommissioned. But third party solutions also need management. To what extent is

maintain them. 'They are of no real use, and potentially a hindrance, if they are not kept up to date at all times,' says Upton. This can be time consuming and typically of a lower priority when compared to other maintenance issues faced by an IT department. Elements can certainly be automated with tools that are available or which can be built, but there are always some aspects which need to be completed, or at least reviewed manually.

When a bank is considering an external product or has to undertake an upgrade programme, introduce a new business process or is forced to integrate with another system as part of a merger or acquisition, its objective should be to understand every impact point of those changes so that it does not inadvertently harm other applications or processes.

The power that this understanding gives is felt most when consideration is given to land-mark decisions such as a change of operating system. Without the background information, such a move could prove catastrophic. As Upton says, 'It's not a question that can be answered over a cup of coffee'. Indeed, much of the time spent on such projects is in assessing their impact. 'Good systems can help a lot,' he says.

When all is said and done, ALM is not a dream solution. 'It probably impacts every single side of the IT process, but it's hard to put a finger on where the big benefit is,' notes Upton. It will not necessarily allow a business to do something it couldn't do before. Without obvious tangible results many see it as just another gimmick or a business overhead to be avoided.

Clearly ALM is the business of both application vendors and internal developers, particularly in the software development field. If we now focus in on the development cycle, does it reveal any further truths?

Upton points out that the lifecycle in this zone rotates many times through the stages of concept, strategy, planning, implementation, validation and delivery, before eventual decommissioning. According to Zhou, this process must therefore yield a 'conceptual product' and an 'actual product'.

Within this process, he says, every development operates on at least two levels. The macro-level covers high-level processes spanning both conceptual and actual prod-

ucts from the informal drafting of an idea to meet a need, the design of formal specifications, the creation of design documentation, development plans, QA testing and release management right through to user documentation. Sitting below this is the micro-level where it is essential for any development programme to consider the assignment of individuals or teams to a task and for them to consider and plan each task in depth.

'The relationship between the micro-level workflow and the macro-level of the lifecycle really enables the development team to control quality of the applications,'



says Zhou. So when development methodologies (such as Agile, Waterfall, CMMI or Iterative) are discussed, the issue really is one of workflow between the two primary lifecycle phases of design and implementation. Each methodology differs in its approach to the interaction of these two key phases, he says. And since each task may be best served by one methodology over another, the effect of deploying more than one methodology within a project may be to destabilise co-ordination and project timing unless tight control is exercised.

But the major problem arises when development is carried out in isolation – which it often is. This can lead to the final release of a multi-part development not matching the exact requirements of the original product specification. Given the need for software companies and in-house developers to be agile, adaptive and quick with their methods, commercial pressure typically dictates that, whilst the conceptual stage is designed by a central core team,

the actual product more often than not is implemented by distributed or even outsourced teams.

The purpose then of an ALM product is to seize hold of these workflows and development approaches and to enable all users to standardise the way in which lifecycles are controlled – regardless of methodology. The point is also to give a 'functional, centralised view into the various lifecycle phases' and ultimate control over who sees what. The end result should be a well-maintained and repeatable process allowing flexibility and control over tasks within single applications and projects that may cover more than one application.

There are, however, a number of issues that can affect ALM, says Zhou. As a particular concern, he cites security and the involvement of vendors when working with third party software, whether at a macro- or micro-level.

How does a bank make documentation available to its own development team, but restrict access to a third party team, when both are closely involved in a project?

'Normally a company might open a web portal for the third party to access its data,' says Zhou. SnapLock Storage and Retrieval (SSAR) software can enable connectivity for the third party but, at a data level, additional security controls may be required, controlling the view that the third party may have.

At the macro development stage, an in-house team might require a third party to develop some components of an internal application. It may be that separate projects have to be created, with an inter-project communications link between all parties. If this is so, it becomes critical that the third party task is well controlled and follows the same workflow examples.

Any ALM product on offer should be designed to enable third party involvement in all phases of the application lifecycle, but only allow them to view and modify the relevant aspects.

Where development is spread across multiple countries, time zones and cultures it is important that control is exercised over certain localised functionality, such as tax. By breaking the project down into its macro-design and micro-execution stages, it will be possible for the design element to take precise control over what the execution team does. 'A product developed by a

bank may need to be pushed to different regions so the ALM product needs to maintain different releases for different countries,' says Zhou.

The problem of how to make the software belong to the same bank and yet offer the requisite variations naturally requires the development team to be capable of carrying out multi-release management with separate codelines for each release in each market – the different codelines corresponding to the different design tasks. From a source code control point-of-view, a bank will need to maintain multiple branches of that code.

To allow access there must be, as an option at least, a centralised database server, web server and application server. 'Taking this approach means all the development issues, all the planning data, all the requirement analysis data, programming tasks, and coding QA testing tasks are centrally saved in one database server with supporting document servers,' says Zhou. For different teams working in different countries, or on different product lines, or variations of the same product, each will only see the portion of the data and documentation that it needs to see.

A user could of course deploy the database and document servers at different sites, relying on controlled communication between each location. This, notes Zhou, would only be advisable where good connections exist between multi-site teams – where sufficient bandwidth can be purchased, something which is not always available.

Setting up an ALM system will usually require a project to be pre-configured – how the lifecycle should be maintained and how the workflow should be controlled. Workflow in this case covers every development task and each of its multiple states. It enables fluid movement through

each state, under the guidance of the pre-set definable actions. 'Once you define the actions, you define the logic to enable your team members to push a development task from a current state to the next possible state.'

Actions in the process must define process automations, the user interface, how each consecutive project should be inter-related, how inter-project communication can happen, how notification of project assignment will be carried out, and the routes for automated escalation where delays are being experienced (normally email warnings or change of task ownership).

'Once the project is discussed, and everyone is satisfied and likes the settings, it becomes a standard for the next time you need to run a similar project,' says Zhou.

Of course, the success of any ALM-led project can be measured by comparing the released actual product to the final state of the conceptual product. Ultimately though, real success will depend largely upon how closely the actual product answers the underlying business need. For that to happen, the ALM process needs to look at the bigger picture.


For the ALM movement to gather pace it needs some real coherence. All the elements that can be controlled need to be linked together. As Upton says, 'there should be a thread that is traceable, right through to the end'.

Currently, as has been stated, this is not possible. However, there is much talk in the industry right now about the concept of ALM 2.0. This for now is just a loose theory, but it has continuity as its goal. IT is being pulled in many directions at once by forces such as collaboration, reuse, business alignment, SOA, regulatory compliance, process reengineering, outsourcing, off-shoring, platform neutrality, open source, standards,

SLAs, Web 2.0, and grid computing. The concept of ALM 2.0 appears to be a platform for development where all these activities can be co-ordinated and managed.

For the time being, Both Upton and Zhou agree that the real benefits of ALM lie in standardisation and efficiency. Upton adds that what a business can do is to handle its infrastructure 'more flexibly'. In theory, ALM should also create a more robust environment because, as part of it, any business will be able to identify and log what IT it has and locate all its interfaces. This makes it far easier to pinpoint bottlenecks and failures. A business with a managed ALM programme is also going to find out how far it can scale its processes. This alone surely helps in deciding the future direction of many aspects from the products and services offered through to infrastructure modelling.

The knock-on effect for a bank's customers is in improved day-to-day running efficiency, less outages from overnight or weekend maintenance, and faster time to market for new and variable products. Surely a bank that has grown up and knows exactly where everything goes is then able to provide a better experience all round.

At the end of the day, isn't it just a case of good housekeeping though? 'It should be,' says Upton. 'It's the same as all good management theories. They're all based on common sense.' Indeed, 'perfectly adequate' ALM can be achieved just by adopting a good housekeeping attitude, he insists. 'Traditionally, businesses that have had good applications maintenance and infrastructure programmes have seen it as part of good housekeeping.' Businesses are free to operate as they choose of course, but without a strict regime of everything-in-its-place, ALM software can at least provide a helping hand. 

Publication: International Banking Systems Journal. Editor: Martin Whybrow.

This article has been extracted from the International Banking Systems Journal. February 2007 issue 16.5

The International Banking Systems Journal is dedicated to the wholesale, retail and private back office banking systems and operations market, and related topics.

IBS Publishing Limited. Publishing office: IBS Publishing Limited, 8 Stade Street, Hythe, Kent, CT21 6BE, UK.

Registered in England and Wales No. 5365737

Tel. +44 1303 262 636 Fax. +44 1303 262 646

Email. enquiries@ibspublishing.com Website. www.ibspublishing.com

© Copyright IBS Publishing 2007 – Material may not be reproduced in any form without the written permission of the publisher.

