

Open Source Support for the Application Lifecycle

ALM integration aims to increase the productivity of software development

Tim Jennings

Increased productivity within the application development function has been a continuing goal, and the current focus is on closer integration between the different phases of the application lifecycle, including requirements management, modelling, development, change management, testing, and deployment. One way to achieve this integration is by selecting a suite from a single vendor, but this can result in poor or missing coverage of some lifecycle areas, and risks vendor lock-in.

The emerging alternative has been the development of open source frameworks such as Eclipse, which allows an organisation to select best-of-breed tools in each area that plug into a common foundation. This latter model has gained significant traction, but only offers integration of tools, rather than of the development process itself. A new initiative under the Eclipse banner, called the Application Lifecycle Framework (ALF), promises to change this, and will add further to the strength of the Eclipse proposition. ALF creates a framework of Web services that represent elements of the application development lifecycle, linked to a common set of metadata, with Business Process Execution Language (BPEL) used to orchestrate the interactions between these services.

Application Lifecycle Management (ALM) involves managing software development as an end-to-end business process, and has significant benefits in terms of higher project success rates, improved quality of the delivered solution, and reduced development timescales. Almost every organisation that undertakes more than just an occasional development project is now focusing on how to apply these principles to their own development teams and management structures, but it is a complex problem requiring a multi-disciplinary approach. From a management perspective efforts will typically concentrate on adherence to defined development processes, such as best practices in project initiation, effective capture of business requirements, design in relation to testing and deployment, and disciplines in creating and maintaining the code base.

Accompanying these trends has been notable improvement in the quality of the tools that are available to support each phase of the software lifecycle. At the start of the process there has been the emergence of Project Portfolio Management (PPM) tools, providing greater visibility into investment and resourcing decisions; requirements capture tools have matured from a point-in-time solution to managing requirements over the whole project lifecycle; modelling tools have become more

widely adopted and offer greater usability; change management tools provide effective control over software assets; testing tools are more comprehensive and are increasingly automated; and there is a wider range of tools to manage application deployment and performance.

Despite these improvements, one significant barrier still exists to adopting this end-to-end perspective across the software lifecycle: the lack of integration between the tools used in each phase of the process. For example, how can the addition of a new requirement automatically be reflected within the development specifications; how can the check-in of a new software module trigger generation of the appropriate test plan; and how should data on application performance be fed back into the design of the next version of the software.

Traditionally, such links, as far as they existed, were purely point-to-point integrations, reflecting market demand or strategic vendor relationships. Thus a vendor of a requirements management solution might offer out-of-the box integrations with the top two or three Integrated Development Environments (IDEs), and possibly also with a Software Configuration Management (SCM) solution. For less common tool combinations, customers either had to request custom integration work from the vendor, or carry out the job themselves. Even where integration did exist, it tended to be at the lowest common denominator level: carried out in batch mode rather than real time, and exchanging information on development artefacts rather than development processes.

As ALM has increased in importance, the past two years has seen an alternative approach from the vendor community, driven both by commercial imperatives and frustration at the technical inadequacies of integration, that has been based on the acquisition of point tools and the subsequent creation of broader ALM suites – IBM, Borland, Serena, and Telelogic all fall into this category. There is no doubt that this ALM suite approach has improved the potential to manage software projects, and has been well received by customers – particularly those who already used several point tools from a particular vendor's line-up. However, the two weaknesses to this approach are firstly that no vendor has been able to establish a uniformly strong offering from end to end, with testing tools being the noticeable area where standalone providers are predominant. Secondly, few organisations operate in an entirely homogeneous environment, so the use of multiple, disconnected tools is the norm.

Consequently, there is a requirement for a

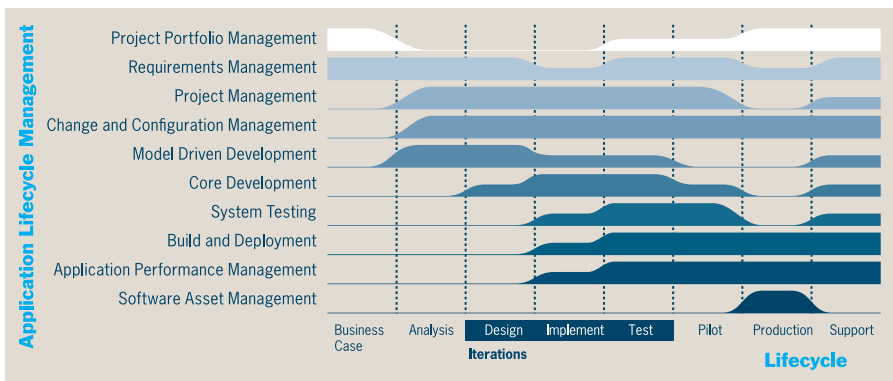


Figure 1:
The Phases of the
Application Lifecycle

standards-based method of integration, which would allow interoperability of tools from multiple vendors. This is, of course, precisely what the Eclipse platform has done for IDEs, and it is from the same source that a potential solution to the ALM problem has emerged, in the form of ALF. The nature of the challenge is somewhat different however – whilst Eclipse has provided a central platform into which development tools and add-ons can be plugged, ALF must support a more complex set of interactions across very diverse categories of software.

In formal terms, ALF is a sub-project of the Eclipse Technology Project, aiming to solve the integration and interoperability challenges of the ALM community (including both users and vendors of ALM tools). It promises to provide an intermediate communication format between lifecycle tools, in such a way that any tool provider need only integrate to this common format, which could then be understood by any other compliant tool. This removes the need for each vendor to create custom integrations with a large number of disparate tools. In addition, ALF proposes a central framework that will manage communication between multiple tools, orchestrating the flow of information into meaningful processes.

ALF is based on a publish-and-subscribe event-driven model, makes use of Web services as the communication protocol between tools and framework, and utilises the BPEL specification for process definition and orchestration within the framework. It consists of three primary elements: the first, ALF Events and Events Vocabulary, define the primary events that might take place within the development lifecycle, such as checking code into a repository, creating a new test plan, or deploying a new software version. It is envisaged that ALF will consist of a basic vocabulary which can be extended for specific scenarios. Secondly, ALF Provider Services and Service Vocabulary will define the services that could be offered by an ALM tool, either to generate or react to an event. Thirdly, ALF Service Flows are the process flows and orchestrations between multiple tools – these are invoked by an ALF Event Manager, in response to a particular event, and allow the framework to trigger actions within the underlying tools.

An example of this might be the creation of a new requirement within a Requirements

Management (RM) tool. The RM tool would trigger an event that is communicated to the ALF framework, which would then invoke a service that includes actions to add that requirement to a skeleton model, and to reflect that requirement within the outline test plan. Similarly, the act of checking-in a completed piece of code would trigger the appropriate actions within testing tools and within a project management tool. It is the vocabularies that will be the key element of ALF, since these will define the metadata that governs the exchange of information between tools, and the nature of the service flows that are created. Vocabularies will be created relating to each lifecycle phase (or subject area in ALF parlance) at two levels – a basic level that will effectively be a lowest common denominator, and an extended level, providing a richer set of information to be exchanged.

The initial ALF subproject has attracted primary support from Serena Software, Catalyst Systems, Cognizant Technology, Compuware, Secure Software, and Segue (now part of Borland), which represents a good cross-section of the development lifecycle. In different ways, two gaps are apparent – firstly the absence of a leading testing tool vendor from the list, and secondly that some of the larger application development players are not participating. I do not believe that either of these is of significant concern, since Eclipse has such a powerful momentum in the market, that there are significant disadvantages to vendors who lag behind in adoption, once user demand has been generated.

The exception of course is Microsoft, and whilst it is highly unlikely that the company would get directly involved, many of the lifecycle management tools do bridge Java and .NET environments. Other companies have also committed to supporting ALF within their products, an interesting example being AccuRev, which provides SCM solutions with a strong process element. AccuRev is typical of the smaller players in this market, who will both contribute their expertise to the ALF subproject, and benefit from the more open, integrated environment that is envisaged.

The one concern that I have over ALF is its reliance on technology such as XML vocabulary-driven Web services and BPEL. Technically these are all logical choices, but the standards are not yet completely stabilised or universally adopted, and there is a risk that the delivery of ALF as a practical proposition, envisaged in phases through the second half of 2006 and into 2007, may be delayed as a result. I sincerely hope that this concern proves unfounded, because ALF will make it notably easier for customers to support their software lifecycle management efforts with either suites or best-of-breed tools. Ultimately this translates into better visibility into the application development function, and allows it to be managed as a core business process. In the May 2006 edition of *Butler Group Review*, Michael Azoff will look at how Agile development methodologies are now being applied to the whole software lifecycle.

At a glance

- Integrated ALM suites from a single vendor are not yet equivalent to best-of-breed offerings across all lifecycle phases.
- Eclipse offers a common foundation for integrating disparate tools, but has not thus far addressed process-level integration.
- The Application Lifecycle Framework (ALF) is a new Eclipse initiative that will use common metadata, Web services, and BPEL to integrate ALM processes.
- The first version of the framework is due for release in 2H, 2006.
- ALF will make it easier for customers to support their software lifecycle management efforts, which translates into better visibility into the application development function.



Tim Jennings
Research Director

tim.jennings@butlergroup.com