TRENDS

August 18, 2006

The Changing Face Of Application Life-Cycle Management

by Carey Schwaber

FORRESTER® Helping Business Thrive On Technology Change

August 18, 2006

The Changing Face Of Application Life-Cycle Management

Tomorrow's ALM Platforms Will Deliver On The Promise Of Today's ALM Suites

by Carey Schwaber
with John R. Rymer and Jacqueline Stone

EXECUTIVE SUMMARY

IT organizations spend billions of dollars a year on development life-cycle tools that don't play well together. The result? For most shops, application life-cycle management (ALM) — the coordination of development life-cycle activities — is still largely a manual process. Today's ALM suites don't offer much support for ALM beyond what can be accomplished through brittle tool-to-tool integrations. But tomorrow's ALM platforms will do much better by providing common services to practitioner tools. These solutions will be easier to implement, maintain, and employ. And at the end of the day, they'll enable development organizations to build better software.

TABLE OF CONTENTS

NOTES & RESOURCES

Forrester interviewed 14 vendor and eight user companies, including Borland Software, Compuware, The Eclipse Foundation, Mercury Interactive, Microsoft, MKS, Oracle, and Serena Software.

Related Research Documents

"Getting Ready For Microsoft's Team System" October 20, 2005, Trends

"Integrate Requirements Management With SCM To Enable More Informed Project Management" September 21, 2005, Quick Take

"The Expanding Purview Of Software Configuration Management" July 22, 2005, Trends

"Lightweight Tool Sets Represent An Alternative To Integrated Tool Suites" April 21, 2005, Trends

TARGET AUDIENCE

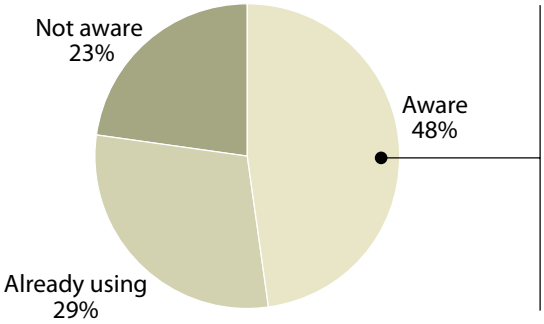
Application development professional

AWARENESSS OF ALM IS HIGH; UNDERSTANDING IS LOW

The good news is that ALM — the coordination of development activities to produce software applications — is an accepted concept. Our data shows higher levels of awareness among IT shops of ALM than of other emerging disciplines like information life-cycle management and Agile software development processes. Nearly one-third of enterprises are already using ALM, and almost half are aware of it (see Figure 1).[1] But in our conversations with user companies, we find that even those familiar with the term are often hard-pressed to define it. This state of affairs — high awareness but low understanding — is the consequence of about a decade's worth of vendor marketing efforts that outpaced vendor offerings.

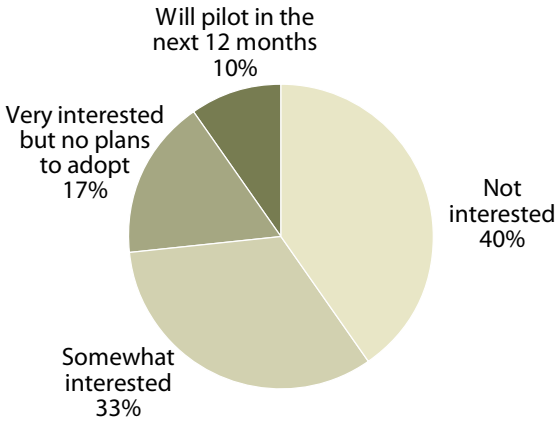
Figure 1 Awareness Of ALM Is High

1-1 "Are you aware of application life-cycle management (ALM)?"



Base: 911 software and services decision-makers at North American and European enterprises

1-2 "How interested are you in adopting ALM?"



Base: 437 software and services decision-makers at North American and European enterprises that are aware of but not already using ALM

Source: Forrester's Business Technographics® November 2005 North American And European Enterprise Software And Services Survey

37653

Source: Forrester Research, Inc.

ALM Is The Thread That Ties The Development Life Cycle Together

Forrester defines ALM as:

> *The coordination of development life-cycle activities, including requirements, modeling, development, build, and testing, through: 1) enforcement of processes that span these activities; 2) management of relationships between development artifacts used or produced by these activities; and 3) reporting on progress of the development effort as a whole.*

Thus, although many think of ALM as simply a product market, it is much more than that. To understand the true scope of ALM, it is crucial to remember that:

- **ALM is a discipline, as well as a product category.** With so many vendors talking about ALM, it's sometimes hard to remember that it can be accomplished without supporting tools. Each of the three pillars of ALM — traceability, process automation, and reporting and analytics — corresponds to a manual process that can be made more efficient and effective through tool integration. For example, one bank told us: "Tracing from use cases to test cases manually using spreadsheets is very labor intensive. That's why we're looking for a requirement management tool to integrate with our test management tool."

- **ALM doesn't support specific life-cycle activities; rather, it keeps them all in sync.** A development effort can still fail miserably even if analysts document business requirements perfectly, architects build flawless models, developers write defect-free code, and testers execute thousands of tests. ALM ensures the coordination of these activities, which keeps practitioners' efforts directed at delivering applications that meet business needs.

- **An ALM solution is the integration of life-cycle tools, not merely a collection thereof.** Effective tool support for ALM connects the practitioner tools within a development project, such as an IDE, a build management tool, and a test management tool. It's the connections, rather than the tools themselves, that make up an ALM solution. As an IT exec at one multichannel retailer put it: "You have to pick tools, obviously. But tools aren't the focus; the focus is how the tools connect."

ALM alone is not enough to guarantee successful software delivery. Much of ALM's value comes from its connections to the separate but related disciplines of project portfolio management (PPM) and IT operations. The best PPM efforts leverage ALM data to inform executive decision-making. And handoffs to and from IT operations fulfill, propel, and inform development activities.[2] The value of these connections corresponds directly to the strength of a shop's ALM practices.

HOW ALM TOOL SUPPORT CAN HELP DEVELOPMENT ORGANIZATIONS

ALM is labor-intensive and error-prone without the integration of life-cycle tools. These connections take ALM to the next level by improving the efficiency of each of the three pillars of ALM:

1. **Traceability of relationships between artifacts.** Correlating life-cycle artifacts like requirements, models, source code, build scripts, and test cases helps demonstrate that the software has delivered functions as the business wanted it to.[3] Internal and external compliance requirements, as well as the increasing need to coordinate development across roles, locations, and organizations, make traceability more of a necessity than an ideal.[4] For most organizations, traceability is a manual process. The problem isn't just the size of projects; it's also the number, the varying size and scope, and the artifact interdependencies. Managing dependencies between high-priority change requests and ongoing application development efforts "sometimes seems like it isn't humanly possible," reports one healthcare company.

2. **Automation of high-level processes.** Development organizations commonly employ paper-based approval processes to control handoffs between functions like analysis and design or build and testing. ALM improves efficiency by automating these handoffs and storing all associated documentation. One financial services firm we spoke with estimated that automating of build-deploy-test processes would save each of its developers an hour a day. Executable process descriptions — process models that correspond to actual automated processes — are a real boon for the many shops that have a "Book of Process" that sits on the shelf and is largely ignored. As one firm put it: "We had a consulting company define a methodology for us. We still have it on a shelf somewhere. A process needs to live in the tools we use if it's ever going to be followed."

3. **Providing visibility into the progress of development efforts.** Most managers have limited visibility into the progress of development projects; what visibility they have is typically gleaned from subjective testimonials rather than from objective data. A bank we spoke with told us: "We do progress reporting the same way we've been doing it for 40 years. It's all manual: weekly status meetings, progress reports, demonstrations. We'd love to get test results from nightly builds posted somewhere instead of having to run people down to ask them whether things are working yet."

ALM 1.0 FALLS SHORT OF THE MARK

The majority of today's ALM solutions have grown through accretion rather than through purposeful design. As a result, the dominant structure of today's ALM solutions is tool-to-tool integration, and this integration is never as deep or resilient as advertised — especially when it's integration of different vendors' tools. ALM 1.0 is characterized by (see Figure 2):

- **A single tool for each role.** The problem with role-based tools is that roles are anything but uniform, varying by company, by business unit, by development team, and even by individual.

When a customer's roles don't match up with the roles for which vendors have built tools, the IT organization has to choose between changing its roles, licensing multiple tools for a single role, or purchasing more features than a given role is likely to need. To provide an audience of diverse practitioners with all the features they need, vendors end up stuffing tools with so many features that they are practically unusable.[5] The result is complex and expensive tools that have more functionality than any one individual is likely to need.

- **Redundant and inconsistent ALM features locked in practitioner tools.** Today's life-cycle tools feature an impressive amount of redundant and usually inconsistent functionality in areas like workflow, collaboration, reporting and analytics, and security. What's worse, functionality that would be valuable to the entire development team is often available only from within a single practitioner tool. IBM Rational RequisitePro, for example, is the only Rational tool with a discussion board. And Borland JBuilder includes innovative collaboration capabilities that aren't accessible from other Borland tools or subject to access restrictions imposed by those tools.[6]

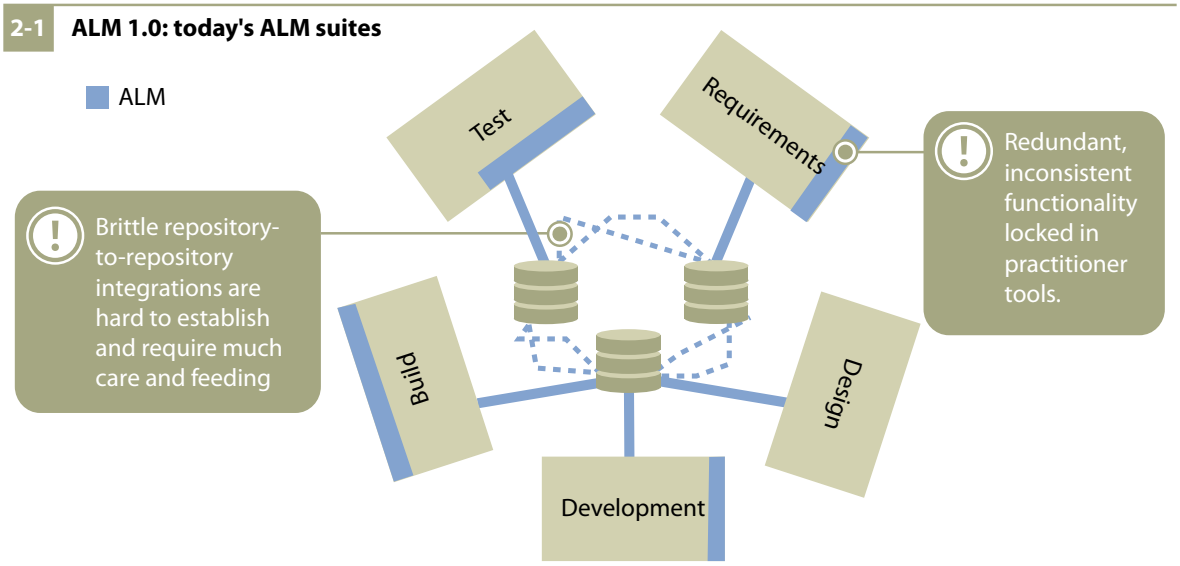
- **Microprocesses embedded in tools and macroprocesses in tool integrations.** In ALM 1.0, the microprocesses that regulate practitioner efforts are embedded in each practitioner tool (e.g., VBScript workflows in Mercury Quality Center), and the macroprocesses that regulate interactions between these practitioners live in the integrations between these tools (e.g., connectors between Serena TeamTrack and Mercury Quality Center). This means that process assets aren't versionable assets, can't share common components, and can't be managed as a portfolio. For this reason, most shops focus their process governance efforts on paper-based process assets, hoping that they correspond to the processes instantiated in their tool sets.

- **Integration through brittle repository synchronization mechanisms.** Repository synchronization is the primary means for integrating life-cycle tools today — even when the tools concerned are all from the same vendor.[7] But it is often difficult to establish, costly to maintain, or flat-out unworkable. An executive at one retail firm told Forrester: "We love our requirements management tool's functionality, but it wasn't an open tool so we're going to have to drop it. It's supposed to have 'open APIs,' but we couldn't get it to interface with our test management tool, and neither could the vendors." Vendors like MKS and Microsoft attempt to skirt this issue by building ALM solutions around a single repository. For many IT organizations, however, moving onto a single repository is a practical impossibility — either because they've invested so much in these repositories or because their development spans so many different platforms.[8]

Forrester estimates that development shops spend upwards of $5 billion on new licenses for development life-cycle tools every year. But all of the architectural characteristics of ALM 1.0 solutions cited above limit the return that user companies can achieve on their investment (see Figure 3).

Figure 2 ALM 1.0 And Its Hidden Costs

2-1 ALM 1.0: today's ALM suites

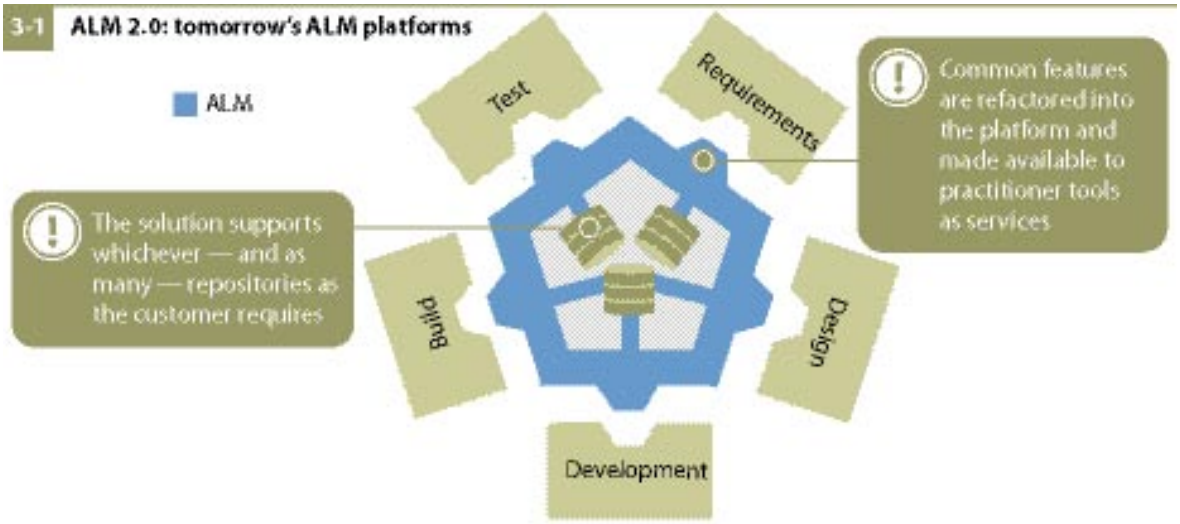


2-2 The hidden costs of ALM 1.0

Characteristic	Cost
A single tool for each role	• Low productivity due to tool complexity • Role misalignment leads to overspending on licensing
Redundant and inconsistent ALM features locked in practitioner tools	• Lack of cross-life-cycle transparency • Ossification of functional silos
Microprocesses embedded in tools and macroprocesses in tool integrations	• Effort spent building and maintaining synchronizations • No single source of truth
Integration through brittle repository synchronization mechanisms	Process assets are opaque and difficult to manage

37653 Source: Forrester Research, Inc.

Figure 3 The Promise And Payoff Of ALM 2.0



3-2 ALM 2.0 is better

Characteristic	Benefit
Practitioner tools assembled out of plug-ins	• Customers pay only for the features they need • Practitioners find the features they need faster
Common services available across practitioner tools	• Easier for vendor to deploy enhancements to shared features • Ensures correspondence of activities across practitioner tools
Repository neutrality	• No need to migrate old assets • Better support for cross-platform development
Use of open integration standards	Easier for customers and partners to build deeper integrations with third-party tools
Microprocesses and macroprocesses governed by externalized workflow	• Processes are versionable assets • Processes can share common components

37653 Source: Forrester Research, Inc.

THERE'S HOPE ON THE HORIZON IN ALM 2.0

Tomorrow's ALM is a platform for the coordination and management of development activities, not a collection of life-cycle tools with locked-in and limited ALM features. These platforms are the result of purposeful design rather than rework following acquisitions. The architectural ingredients of ALM 2.0 are:

- **Practitioner tools assembled out of plug-ins.** An à la carte approach to product packaging provides customers with simpler, cheaper tools. Far from being a pipe dream, this approach is a reality today. IBM has done the most to exploit this concept, currently providing many different grades of development and modeling tools that are all available as perspectives in Eclipse, as well as the ability to install only selected features packs in each of these tools.[9] This approach has not yet been successfully applied outside of development and modeling tools. For example,

today, customers must choose between defect management that's too tightly coupled with test management and software configuration management (SCM) or defect management in a standalone tool.

- **Common services available across practitioner tools.** Vendors are identifying features that should be available from within multiple practitioner tools — notably, collaboration, workflow, security, and reporting and analytics — and driving them into the ALM platform. Telelogic has started to make progress on this front for administrative functionality like licensing and installation. Microsoft has gone even further: Visual Studio Team System leverages SharePoint Server for collaboration and Active Directory for authentication, and because it uses SQL Server as its data store, it can leverage SQL Server Analysis Services and SQL Server Report Builder for reporting and analytics.

- **Repository neutrality.** At the center of ALM 2.0 sits not one repository but many. Instead of requiring use of the vendor's SCM solution for storage of all life-cycle assets, tomorrow's ALM will be truly repository-neutral, with close to functional parity, no matter where assets reside. IBM, for example, has announced that in coming years, its ALM solution will integrate with a wide variety of repositories, including open source version control tools like Concurrent Versions System (CVS) and Subversion. This will drive down ALM implementation costs by removing the need to migrate assets — a major obstacle for many shops — and will bring development on mainframe, midrange, and distributed platforms into the same ALM fold.

- **Use of open integration standards.** Two means of integration — use of Web services APIs and use of industry standards for integration — will ease and deepen integration between a single vendor's tools, as well as between its tools and third-party tools. Many vendors still don't offer Web-services-based APIs, but this will change with time. In addition, new standards for life-cycle integration, including Eclipse projects like the Test and Performance Tools Project (TPTP) and Mylar, promise to simplify tools integration. One case in point: SPI Dynamics reports that its integration with IBM Rational ClearQuest Test Manager took one-third of the time it would have taken if both tools didn't leverage TPTP.

- **Microprocesses and macroprocesses governed by externalized workflow.** The ability to create and manage executable application development process descriptions is one of the big wins for ALM 2.0. When processes are stored in readable formats like XML files, they can be versioned, audited, and reported upon. This facilitates incremental process improvement efforts and the application of common process components across otherwise discrete processes. For example, Microsoft Visual Studio Team System process templates are implemented in XML and contain work-item-type definitions, permissions, project structure, a project portal, and a version control structure.

There is no solution on the market that possesses all of these characteristics, but this is the direction in which most vendors are moving. However, it will be at least two years before any vendor offers a solution that truly fulfills the vision of ALM 2.0.

SINGLE-VENDOR PLATFORMS DOMINATE, BUT MULTIVENDOR PLATFORMS EMERGE

Every major vendor is taking concrete steps toward delivering on the vision of ALM 2.0. But their strategies vary (see Figure 4):

- **Borland and IBM build ALM platforms to tie together existing tools.** Borland and IBM are the two vendors with the greatest breadth of tools support for development life-cycle activities. They've worked long and hard to improve the integration of their various tools through traditional ALM 1.0 means. But their new strategy is much closer to ALM 2.0: ALM platforms that provide common services to more practitioner tools. Both vendors' solutions will be distinguished by their repository neutrality, by the increasing modularity of their practitioner tools, and by support for executable descriptions of some microprocesses but not macroprocesses.

- **Microsoft and MKS build ALM solutions from the ground up around a single repository.** Microsoft and MKS have both built ALM solutions that are oriented around a single repository. The three pillars of ALM — traceability, process automation, and reporting and analytics — are all easier to support than a single-repository solution. But this only shifts the burden onto development shops, requiring them to either migrate all of their assets into the ALM solution's single repository or else be content with limited ALM functionality for assets that remain in place.

- **Serena and Compuware lead efforts to build an open, multivendor ALM platform.** Not all vendors are taking a "go at it alone" approach to ALM. The most notable collaborative efforts in this area are the Eclipse Application Lifecycle Framework (ALF) project and the Eclipse Corona project, which are led by Serena Software and Compuware and joined by smaller vendors like AccuRev and Catalyst Systems. Participating vendors look to these projects to improve their integrations with third-party life-cycle tools, even while they improve the support for ALM available from their own products. Serena, for example, is the lead vendor behind ALF, but it has also moved its requirements management and process-centric SCM solution onto a single repository.

It's clear that most major vendor are building proprietary ALM platforms that will work better with their own tools than with competing vendors' tools. Some services, for example, will be available only to the vendor's own practitioner tools, even if other vendors are willing and able to take advantage of them. Microsoft Visual Studio Team System is an early sign of things to come: Third-party vendors can build integrations between their own tools and Team System, but the depth of integration that can be achieved doesn't compare to the integration in Team System itself.

Figure 4 Major Vendors' Approaches To ALM

Vendor	Approach
Borland	Single-vendor ALM platform
CA	SCM, not ALM
Compuware	Multivendor ALM platform
IBM	Single-vendor ALM platform
Mercury	More test management than ALM
MKS	Single-vendor, single-repository ALM platform
Microsoft	Single-vendor, single-repository ALM platform
Serena	Single-vendor, single-repository ALM platform; multivendor ALM platform
Telelogic	Single-vendor ALM platform

37653 Source: Forrester Research, Inc.

RECOMMENDATIONS

START LOOKING FOR ALM 2.0 TODAY

Every vendor's evolution toward ALM 2.0 will proceed at a different speed and will be guided by a different set of priorities. Evaluate vendors on the basis of their progress from ALM 1.0 to 2.0 by asking them the following questions:

- **How is integration accomplished in your solution?** When user companies ask about tool integrations, they too often accept yes or no answers. To determine how well and how easily a vendor's tools really integrate with each other and with third-party tools, inquire about the technologies used to accomplish this integration. A multichannel retailer that's had great success building out an integrated suite of tools from multiple vendors goes so far as to quiz customer references about integrations, talk to the vendor's engineering staff, and even start working the APIs themselves, if they're available.
- **Can we get just the features we want?** If you're after a certain vendor's requirements definition features but you already have a requirements management solution, seek to pay less for your license. Smart vendors let client demand drive product packaging, so even if the answer is no, your question may have an effect in the long term. If the vendor does offer different modules, be sure to find out how difficult it would be to implement additional modules at a later date. Ask whether this would rip-and-replace the whole system or whether it would merely be a matter of awakening dormant functionality using a new license key.
- **Where is my development process instantiated?** Customers should ask vendors how and where development processes would be instantiated, from requirements all the way through to deployment and potentially beyond. Look at a live connection between process models and the means of process automation — that is, for executable process descriptions. The benefits of process modeling capabilities are far lower if the assets you create can't actually be implemented.

- **Can you do a live pilot for me?** The best proof is the proof of concept. ALM solutions that take lots of time and money to implement tend to also take lots of time and money to maintain. Take it as a bad sign when a vendor isn't able or willing to do a live pilot.

WHAT IT MEANS

THE CHOICE BETWEEN INTEGRATED AND BEST-OF-BREED TOOLS WON'T GO AWAY

Proprietary platforms will have the edge in terms of depth of integration, and this better integration will in turn enable better support for ALM. This means that the choice between best-of-breed tools and tool integration won't go away. User companies will still have to choose between standardizing on a single vendor to get the best ALM capabilities or else sacrificing on ALM to be able to pick and choose each practitioner tool on its own merits.[10] Here's how they'll choose:

- **Shops with mainstream tool needs will tend toward proprietary ALM platforms.** For the majority of IT organizations, life-cycle tools from mainstream vendors are a close fit. For these shops, the benefits of using life-cycle tools that work with proprietary ALM platforms outweigh the benefits of using best-of-breed tools. And as vendors like Borland and IBM break down monolithic practitioner tools into modular feature packs, the chances that those tools will suit a development shop's needs will only increase.
- **Some shops with niche tool needs will turn to niche proprietary ALM platforms.** Some user companies will find value in a single-vendor ALM platform specifically built to suit their style of development — for example, Rally Software Development's ALM solution for shops using Agile processes or VA Software's ALM solution for geographically distributed development teams.[11] The limiting factor on the success of these vendors? The extent to which they resist the lure of the broader market and focus on serving their niche.
- **Shops with unique tool needs will look to multivendor ALM platforms.** Development shops whose needs aren't met by mainstream vendors or by specialist vendors will have to look to open, multivendor ALM platforms like the Eclipse ALF and Corona projects. Such projects represent the best way for these shops to secure some tool support for ALM without giving up the freedom to choose the tools that best suit their purposes.

SUPPLEMENTAL MATERIAL

Companies Interviewed For This Document

Borland Software	Microsoft
Compuware	MKS
Eclipse Application Lifecycle Framework	Oracle
Eclipse Foundation	Rally Software Development
Eclipse Mylar Technology Project	Serena Software
IBM	Telelogic
Mercury Interactive	VA Software

ENDNOTES

[1] This survey research, however, did not differentiate between ALM tools and ALM processes.

[2] Handoffs between ALM and deployment tools fulfill development efforts; handoffs between service desk solutions and issue management solutions propel development efforts; and handoffs between performance monitoring tools and performance modeling tools help guide architectural efforts.

[3] Traceability is based upon a series of assertions. The act of linking a requirement to a test case is an assertion that if the test case passes, then the requirement is fulfilled. While there is room for error in efforts to establish traceability, there's still value in making the attempt.

[4] Regulatory requirements make traceability more than just an ideal: Enterprises now find that they both want and need to confirm that their systems actually do what they've been designed to do. Traceability enables firms to establish this by linking development artifacts like requirements, code, and test cases together and ensuring their correspondence. See the July 19, 2005, Quick Take "Software Configuration Management Tools Ease The Burden Of Compliance."

[5] When vendors attempt to serve too many different types of users with a single tool, the tools in question end up being "bloatware." The most full-featured tool is not always the most successful tool; witness, for example, the decline of Borland JBuilder. This was not due entirely to the ascent of the free Eclipse IDE; after all, Java IDEs like IntelliJ have managed to thrive alongside Eclipse. The popularity of Eclipse is due in part to how easily it enables users to employ plug-ins, assembling the IDE best-suited to their particular needs. See the October 7, 2004, Trends "Eclipse Changes The Game For Development Tools."

[6] Borland JBuilder 2006 introduced the capability for two developers in different locations to work simultaneously on the same source code. They can exchange write locks on the source, see each other's changes in real time, and accept or reject proposed changes. This immediately raises the following question: Wouldn't two designers want exactly the same capability to share models, or two project managers to review plans? Because the feature is embedded in JBuilder rather than the underlying platform, it also raises

governance issues, such as bypassing access controls in the SCM system. For example, a developer who does not have permission access to the module in question through SCM should not be allowed to modify it through the back door of being granted shared access in the IDE by a developer who does have permission.

[7] Life-cycle tool integrations tend to take two primary forms: "on the glass" integration and integration through repository synchronizations. The latter is far more desirable than the former. Repository synchronization itself comes in several flavors: import/export of data from one repository to another, unidirectional synchronization, and bidirectional synchronization. Repository synchronization is rarely available out of the box, since it requires mapping of fields.

[8] Enterprise IT shops often find the cost of migrating assets into a new repository to be prohibitive. In some cases, the investment they've made in this repository is simply too great. This is often the case for SCM, defect management, and test management tools. In other cases, the new repository may lack support for a particular type of asset (e.g., packaged applications) or platform (e.g., Unix, Linux, iSeries, zSeries).

[9] IBM Rational offers standalone modeling capabilities (Rational Software Modeler); these modeling capabilities are also available as part of its IDE for systems development (Rational System Developer). Similarly, Rational offers a lighter-weight Web development IDE (Rational Web Developer) and a more full-featured IDE for J2EE/EJB and portal development (Rational Application Developer). Customers can also purchase Rational Software Architect, which includes all of the capabilities of all of the aforementioned tools.

[10] "Best-of-breed" is something of a misnomer in this context. The challenge for IT organizations is to identify the tool that best suits their particular needs, not the tool that is the absolute best on the market. In some cases, the tool that best meets an IT organization's needs is not the best in its breed.

[11] Globally distributed development spans multiple locations, organizations, or cultures. See the October 24, 2005, Trends "Globally Distributed Development Defined."

FORRESTER®

Helping Business Thrive On Technology Change

Headquarters

Forrester Research, Inc.
400 Technology Square
Cambridge, MA 02139 USA
Tel: +1 617/613-6000
Fax: +1 617/613-5000
Email: forrester@forrester.com
Nasdaq symbol: FORR
www.forrester.com

Research and Sales Offices

Australia
Brazil
Canada
Denmark
France
Germany
Hong Kong
India
Israel
Japan
Korea
The Netherlands
Switzerland
United Kingdom
United States

For a complete list of worldwide locations, visit www.forrester.com/about.

Forrester Research (Nasdaq: FORR) is an independent technology and market research company that provides pragmatic and forward-thinking advice about technology's impact on business and consumers. For 22 years, Forrester has been a thought leader and trusted advisor, helping global clients lead in their markets through its research, consulting, events, and peer-to-peer executive programs. For more information, visit www.forrester.com.

FORRESTER®